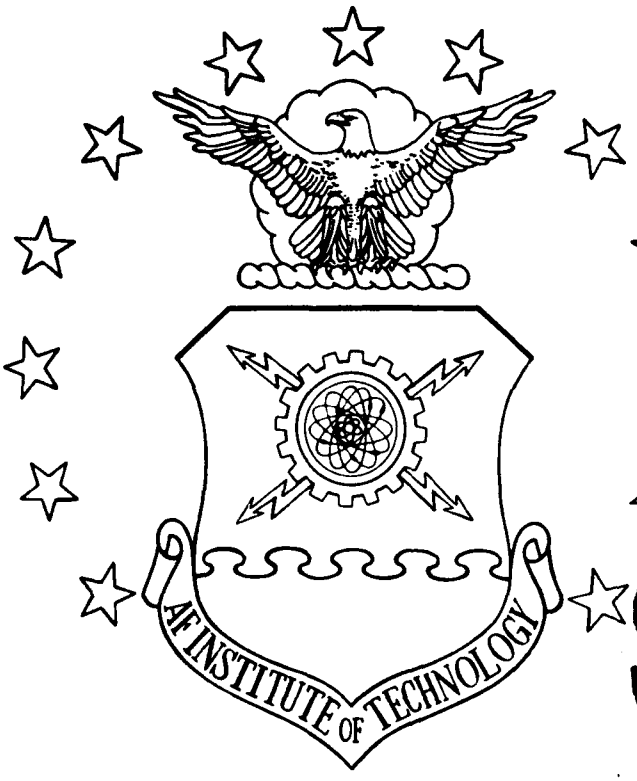
AD-A229 532

(2)



DTIC
ELECTE
DEC 19 1990
S D
D

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

(2)

AFIT/GCA/LSQ/90S-1

DTIC
ELECTE
DEC 19 1990
S D
D

A COMPARISON OF SOFTWARE SCHEDULE ESTIMATORS

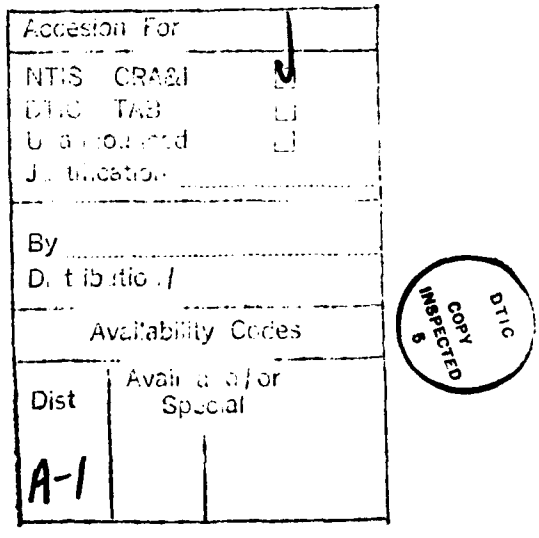
THESIS

Bryan A. Daly, B. S.
Captain, USAF

AFIT/GCA/LSQ/90S-1

Approved for public release; distribution unlimited

The opinions and conclusions in this paper are those of the author and are not intended to represent the official position of the DOD, USAF, or any other government agency.



AFIT/GCA/LSQ/90S-1

A COMPARISON OF SOFTWARE SCHEDULE ESTIMATORS

THESIS

Presented to the Faculty of the School of Systems and Logistics

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the Degree of

Master of Science in Cost Analysis

Bryan A. Daly, B. S.

Captain, USAF

September 1990

Approved for public release, distribution unlimited

Acknowledgements

This research effort has been a very rigorous endeavor which would not have been completed without the aid of key individuals. I would like to acknowledge those people now.

I would like to thank Ms. Peggy Wells at the Electronic Systems Division for assisting me in finding a good data base with which to work. I would also like to thank Mr. Paul Funch for supplying me with that data base and indicating certain problems in analysis of this type of data.

I am thankful to Professor Daniel Ferens in supplying the models used and in contributing his valuable experience to the project. I would also like to thank Dr. Roland Kankey for being my thesis reader, giving me much needed feedback and guidance.

I would be remiss in not thanking my classmates for their support and aid in completing this enterprise. I would particularly like to thank De Weelborg McMurry and Dimitri Yallourakis for helping me to maintain my perspective and drive.

I would most like to thank my wife Susan for putting up with me over the last 15 months. There have been many joys and sorrows incidental to this venture through which she has greatly assisted me. Only through such warmth, love, and support have I accomplished what I have.

Table of Contents

Page

List of Figures

List of Tables

List of Equations

Abstract

Accurate schedule estimation in software development programs is important because schedule is a major determinant of cost. Further, as a greater percentage of weapon system cost is taken by software, there is a greater need for knowledge in this area.

In order to verify the accuracy of schedule prediction for software development obtainable today, this effort examined five commercially available software cost/schedule estimators. The estimated results were analyzed for their accuracy in predicting the actual schedules experienced on the projects. The models analyzed were REVIC, PRICE-S, System-4, SPQR/20, and SEER.

A COMPARISON OF SOFTWARE SCHEDULE ESTIMATORS

I. Introduction

General Issue

Development of quality software has become increasingly more important to the Department of Defense. Software costs in the DoD totaled $ 3.3 billion in 1974 and rose to $ 10 billion in 1984, a 12% average annual increase in cost (6:1462).

Specific Problem

The dilemma of developing quality software for the Air Force is often specifically related to the funds available. Air Force managers are continually required to execute programs within a specified budget. This means that we are constantly attempting to generate highly accurate cost estimates for a project so that adequate funds can be programmed and budgeted to develop the required software.

A major cost driver for software development is the development schedule (4:466-467). One author "was struck by the fact that estimates of developing software program costs (effort) were relatively better than those of the development calendar times" (40:51). One expert feels that software

schedule estimation is the "third wave; the next major area of emphasis in software estimation", the first two being software cost and size estimation (14:41). If the schedule for the development of the software can be accurately forecasted, the Systems Program Office will have a greater probability of an accurate estimate of the funds required to develop reliable software that is adequate for the intended task. Given the difficulties of estimating software schedule, what computer programs give the best estimates?

Assumptions

The following assumptions have been made for this research effort:

1) Historical data including estimated versus actual schedule of software development programs is available from some organization within the Air Force.

2) The literature available regarding software development cost models is sufficiently detailed to allow comparisons among the model results.

Research Objectives

In order to fully investigate and secure a significant conclusion to the research problem, the following research objectives were achieved:

1) Gain sufficient knowledge on the subject of software development to understand the estimating process. Particularly, gain knowledge in

the area of software schedule estimation to be qualified to assess the accuracy the different models exhibit;

2) Develop adequate expertise in th- models used so that estimates may contain minimal error and be analyzable.

Investigative Questions

The following investigative questions are raised to support the research objectives:

1) What are the factors that affect a software development program schedule?

2) What general methods are available to generate a software schedule?

3) What computer based programs are available to project the schedule required for a software development program?

4) What methods are available to measure the accuracy of the program's estimates, statistical or non-statistical?

5) Using the selected programs and accuracy measures, which techniques rate the best?

Scope and Limitations

The following limitations will define the scope of this research effort:

1) Only five current software development cost models were chosen for analysis.

2) The selection of the models depended on the availability of thorough documentation for each model, the availability of the model itself and the ability to access the model to run the data.

3) The data base employed contains data of limited breadth. In some cases a project (observation) will be missing a measurement that is required to run the model.

II. Literature Review

The field of software development is of interest to a wide variety of people from the programmers themselves to the managers of the largest corporations, including the Air Force and DoD. An Air Force Studies Board stated that:

> Developing reliable software that is able to perform its intended function has been a problem since the advent of the digital computer. The Commander of the Air Force Systems Command states that it is now the most serious obstacle to systems development. (1:5)

From this great interest it is easy to assume that there is a great deal of information available in the general domain of software development cost estimation. While this is generally true, there is relatively little information that specifically addresses the issue of software schedule estimation.

The purpose of this chapter will be to address current views on software development cost estimating and examine the state of software schedule estimation today. A review of the Department of Defense Standard directing the development of software (DoD-STD-2167A) will be presented, and cost models that will be used in this research effort will be discussed.

Software Development Life Cycle

Authors have broken the software development life cycle into its major stages, each author seeing the life cycle somewhat differently. Most

of the authors reviewed had included at least four main steps in the life cycle: 1) Definition, consisting of feasibility studies and requirements definition; 2) General Design; 3) Implementation, entailing actual program coding, module level testing, system level testing, and system installation; and 4) Maintenance (22:25; 41:14-16; 8:4-6; 4:36-37). This general model may be said to follow the traditional Waterfall model of software development first promoted by Winston W. Royce.

The basic Waterfall model as advocated by Royce uses built in feedback. He states that as each step progresses and the design is further detailed, there is an iteration with the preceding and succeeding steps (39:1). The change process is limited by not looking any further than the surrounding steps (or rarely doing so).

Examples of other software development models include Rapid Prototyping, Incremental Development, Boehm's Spiral Model, Reusable Software and the Transform Model (3:41; 10:1453-1454).

The early stages of the software life cycle are very important to the cost and quality of the end product. Studies have shown that most errors occur in the Definition and Design stages of the life cycle but the errors are not discovered until the Maintenance stage (8:6). Based on a study of major firms' large projects, Boehm estimates that it is up to 100 times more costly to correct an error during the Maintenance stage than it is during the Definition or Design stages (4:39-40).

DoD Standard 2167A

DoD Standard (DoD-STD) 2167A, "Defense System Software Development," "establishes uniform requirements for software development that are applicable throughout the system life cycle. The requirements of this standard provide the basis for Government insight into a contractor's software development, testing, and evaluation effort" (11:iii-iv).

The standard does not specify or discourage any particular method of software development, but does require that the major software development life cycle activities listed in Table 1 be included (11:iv,9).

Table 1
DoD-STD 2167A Life Cycle Activities

1) System Requirements Analysis/Design
2) Software Requirements Analysis
3) Preliminary Design
4) Detailed Design
5) Coding and CSU Testing
6) CSC Integration and Testing
7) CSCI Testing
8) System Integration and Testing

The standard further states that the activities listed above may overlap, be accomplished iteratively, or may be accomplished recursively (11:9).

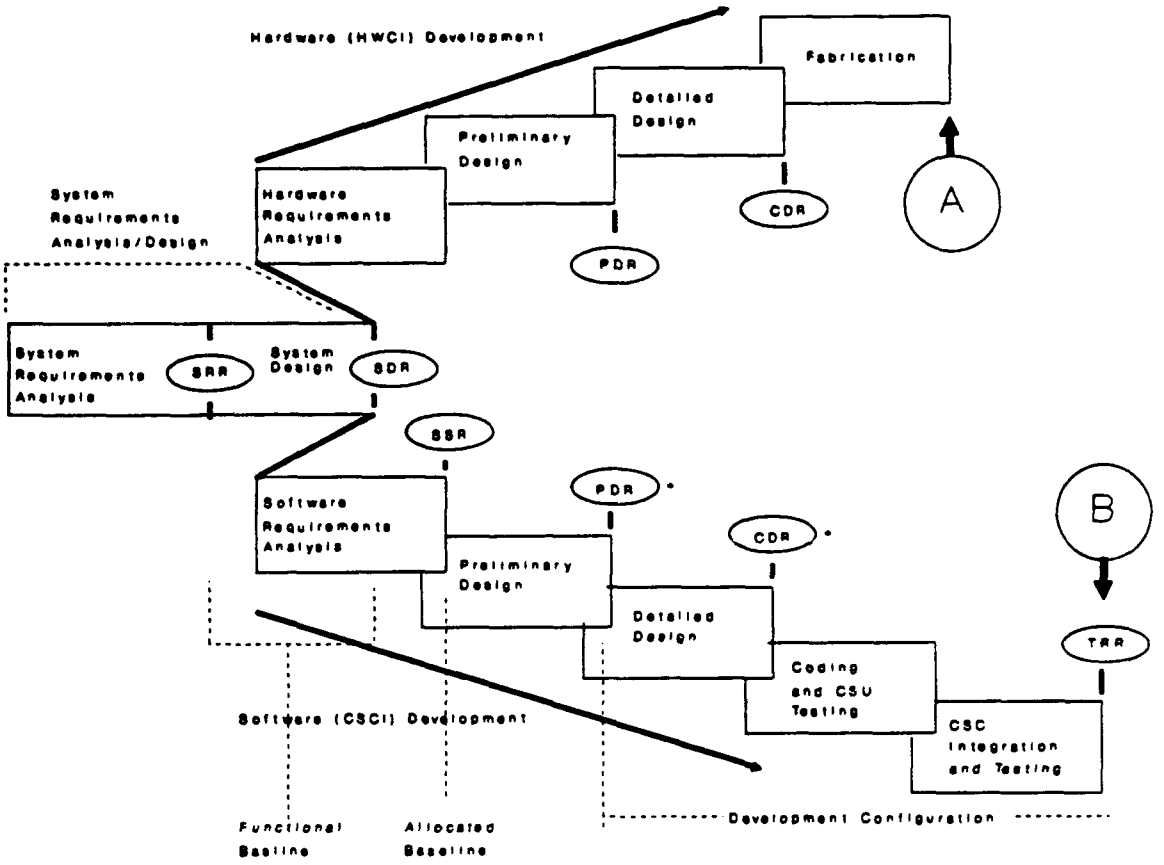
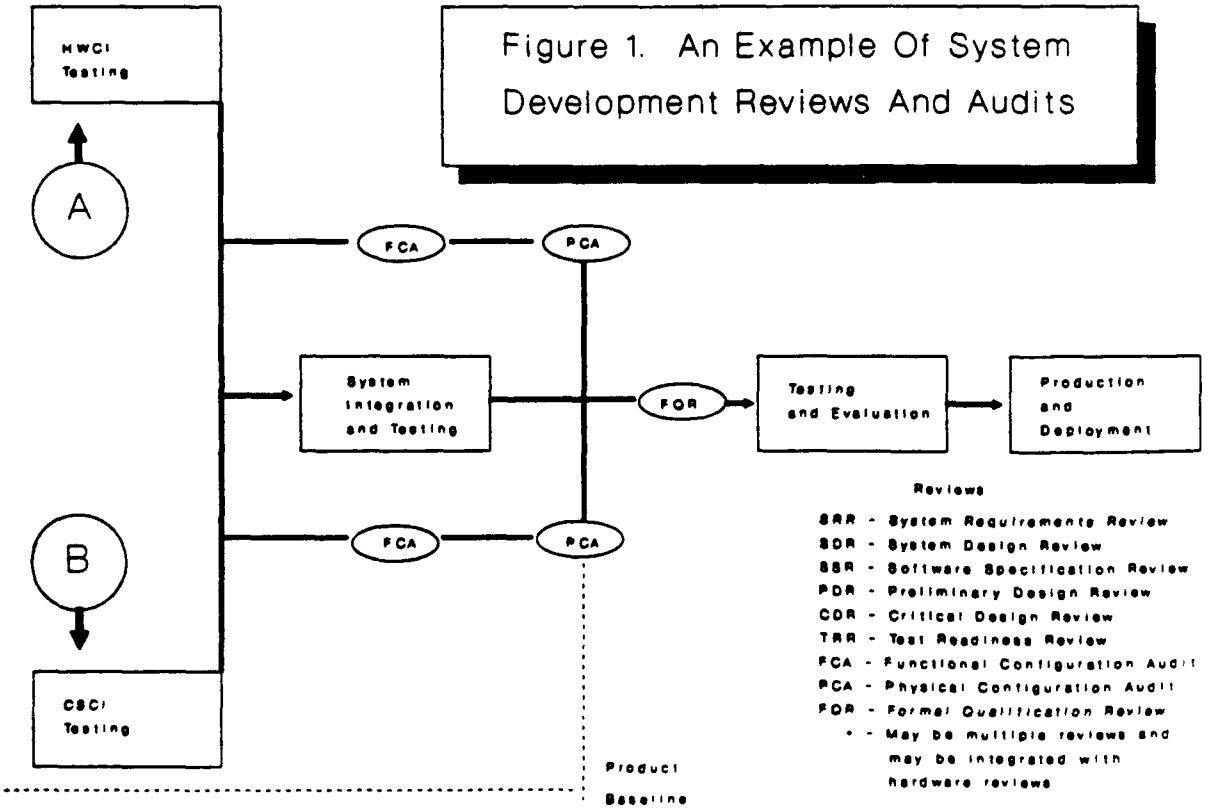


Figure 1. An Example Of System Development Reviews And Audits



DoD-STD-2167A also lists the required software development reviews and audits (see Figure 1). As can be seen from this figure and the required software life cycle activities listed above, the standard uses the software development life cycle traditionally known as the waterfall model.

While the standard may state that no particular development method is supported or discouraged, at least one expert disagrees. Firesmith, in examining the use of DoD-STD-2167 (the precursor to DoD-STD-2167A), looked at the applicability of the Waterfall Method to Ada programming and modern design practices, and did not like what he found. Among the problems are:

1) Artificial Intelligence and Object Oriented Programming can not be mapped onto a waterfall structure (15:56).

2) The compilation order restrictions of Ada encourage a bottom-up approach to software design, while the waterfall method promotes a top-down approach (15:56).

3) The linear nature of the classical life cycle, with formal review bottlenecks between phases prohibits the use of recursive methods (such as Object Oriented Design) (15:57).

4) The lack of useful software to test until the end of the life cycle may cause some managers to rush into coding too early in order to satisfy the users (15:58).

5) The model does not adequately address the reuse of software (a major goal of Ada), or the methods for automatically developing software documentation (15:58).

6) With Ada, coding the specification is frequently a design activity, blurring the distinction of the different phases (15:58).

7) On large projects, the long delay between requirements definition and implementation may combine with personnel turnover to cause changes in the project (15:58).

Software Development Environment

Another area of concern is that of the software development environment. Uncertainties in the software development environment affect the cost of that software. These uncertainties include technical feasibility, experience of the personnel, time constraints, quality tradeoffs, and other goal tradeoffs (18:76).

Technical Feasibility. Software programs are being required to do more than they had to in the past. System architects are forcing software developers to compensate for inadequacies in the hardware by adding more functions to the software. At the same time hardware manufacturers are trying to reduce the size and energy requirements of the hardware, further confining software coders (21:40-41). The extra requirements placed on the software are driving up the costs of software development because of increased technical risk.

Parikh agrees that there are technical feasibility problems with software development. He, however, believes that the technical problems are due to software productivity gains not keeping up with changes in the hardware (35:143). Hardware productivity gains are occurring often enough that software is having a hard time keeping pace. This is illustrated by the release of the 80386 micro-processor by Intel Corporation before the development of an operating system for the 80286 micro-processor.

Being constrained by the hardware of a system is the most difficult problem to overcome because it is the least controllable by the software manager. Using software tools to help in the programming could be a great productivity booster (4:678). Parikh supports this view stating ". . . software tools, provide a means of automating many processes . . . thereby improving human productivity (35:143)." Tools available include integrated debugging environments, compilers, pre-compilers, and Computer Aided Software Engineering (CASE) tools. All can aid in software development.

Personnel Issues. As long ago as 1968, researchers found individual differences in programmer efficiencies of up to 28-1 in debugging, 25-1 in capability to code, 11-1 in programming timing efficiency, and 6-1 in sizing efficiency (21:11). While it is clear that there is a significant difference in programmers' abilities, companies will generally emphasize the bottom line in their cost control efforts, hiring less skilled programmers to save money

(4:669). By doing so, they may actually be harming their ability to finish a quality program on time and at cost.

The difference in productivity between a poorly trained programmer and a well trained programmer is so marked it would do the software manager well to emphasize this area. Studies show the productivity gains achieved by hiring a higher level, better trained programmer are usually well worth the extra pay it costs (6:1465). It is unlikely that there will be a factor of six difference in pay between a top level programmer and one who is below average, offsetting the minimum 6-1 efficiency increase expected from hiring the more expensive programmer (21:11).

Another factor affecting Personnel Experience is the life cycle of a programming team. By looking at this life cycle a software manager can gain insight into possible areas on which to concentrate attention. The cycle is composed of recruitment, goal setting and organizing, doing the work, and dissolution or acceptance of new goals (43:85).

Recruitment. Recruitment is the formation of the software programming team. In this area concentrate on finding the right people for the team. This may be done by using the staffing principles outlined by Boehm. They consist of 1) The Principle of Top Talent, 2) The Principle of Job Matching, 3) The Principle of Career Progression, 4) The Principle of Team Balance, and 5) The Principle of Phaseout (4:667).

Top Talent. This principle states that it is to the advantage of the software manager to hire the best programmers for the job, regardless of the cost. A study by Augustine in 1979, cited by Boehm, showed that the top 20% of performers in a given field produce 50% of the output, and the bottom 50% of the people produce 20% of the output (4:668). By paying for the top talent the software manager will get that top 50% of output, therefore needing fewer programmers on the team. This will save money for the developer. This further supports the point made by Glass regarding Personnel Experience enumerated above.

Job Matching. Job matching requires fitting the tasks to the skills and motivation of the people available. One technique to be used is realistic job preview, in which the software manager realistically states the duties and behaviors expected of the programmer (27). Realistic job preview allows the programmer to self select or self eliminate for a job opening.

Career Progression. Boehm believes that career progression is the way programmers self-actualize (4:670). An example of a situation in which this principle is not followed is software maintenance. A programmer in this area of expertise frequently remains on a project for a long period of time. Remaining on the project provides continuity for the project but forces the individual to program in only the language in which the original project was coded. Further, a software manager will

usually not send this person to symposiums or conferences because the need is not "job related". The programmer finds this situation intolerable and quits, making the situation on the project worse than what the manager was trying to prevent. Filling the need for career progression supplies one of the growth needs cited by Alderfer (9:97). The importance of maintaining career progression is further supported by Metzger, an authority on software project management (33:130).

Team Balance. The team balance principle requires that you "select people who will complement and harmonize with each other (4:671)." This principle not only applies to balancing technical skills on the team but also applies to balancing the psychological components of the team.

Phaseout. This principle states that no matter how careful a software manager is in the selection process, some programmers will not really fit in the team. If this is the case, the programmers must be removed from the team. It is not to anyone's advantage to allow the programmer to remain with the team if other members must do the individual's work. The software manager, however, must remember not to solely base the evaluation on code output. Other contributions to the team, for instance a person who can evaluate the needs of the user well or who is a major contributor to morale, can be just as important as the more traditional measures of output (4:671-672).

Goal Setting and Organizing. Weinberg states "To achieve true consensus on group goals, there is no better method than having the group set the goal itself (33:76)." This is a restatement of the belief that participative management is a superior method of managing a new project with no set objectives (9:416-418). By allowing the programming team to set its own goals, the objectives become more important to the individuals because they are a reflection of their own feelings.

Doing the Work. This would seem to be the easiest area for the software manager to control, but the manager must not allow complacency to set in. The manager must use proper software management techniques to make certain that the product composed by the programming team fills the need of the user. Metzger frames a four step process to manage the software project: 1) meet the real customer, 2) pick the customer's brains, 3) write it down, and 4) get it approved - gradually.

Meet the Real Customer. A software developer will usually meet with a representative of the user, who will lay out the requirements to be met in the program. The possible problem here is that the developer will be coordinating with a marketing representative or with a contracting person. This in no way indicates what the true operator of the system wants. The software manager must meet the real user and be positive that the needs of the user are understood and met.

Pick the Customer's Brains. Invariably the user will have some idea of how he wants the program done, whether it is in the layout of the data input screens or the general flow of the program for ease of use. The manager must try to incorporate as many of these desires in the finished product as is possible. This is the way to satisfy the user.

Write It Down. Many problems can be circumvented by writing down the needs of the user as understood by the manager. Writing down the needs makes them more concrete, and by writing them down, the manager must think the ideas and needs of the user through. This starts the development process off well.

Get It Approved - Gradually. The software manager must not allow the team to charge off into the programming without getting approval from the user at various steps that what is being coded is what is needed. If the manager waits until the end of the project to verify the product is correct, he could have wasted the entire time on a non-responsive program.

Dissolution or Acceptance of New Goals. This part of the life cycle of a programming team may be looked on as the step of transformational leadership. The project is complete and the programming team must now find new goals (a new project) or dissolve. A good transformational leader will see a vision of where he wants to go (what project to

work on next) and promote that vision to the team (27). The process then starts again.

Time Constraints. Software behaves like any other product with regard to time constraints. If the schedule is too ambitious, the product will cost more. Boehm states that a 25% decrease in nominal schedule time will cost an additional 23% in cost (4:679). Software managers must emphasize realistic schedules for the software development life cycle if they are to complete the project within cost. This is particularly important in government acquisition.

To prevent an unnecessary increase in the cost of software development the software manager must maintain a reasonable schedule. By postponing any additions to the software until they can be added during the regularly scheduled updates, the software manager prevents ripple effects from software development problems from flowing through the initial program development (6:1465). This ripple effect also influences the programming team. For example, adding a task may require reassigning a team member which then requires a change in priorities and manpower, affecting the schedule (12:72).

Quality Tradeoffs. Interactions between reliability and costs force the software manager to make decisions. Reducing software cost at the expense of quality during the development phase of the software life cycle could increase the cost of the operations cycle (5:34). Also, any change in

quality requirements may increase the time spent fixing problems in the code. As Gilb relates, "A little change in quality, might well provoke a calamity (20:81)."

Quality tradeoffs can be minimized by using quality assurance throughout the development life of the software, rather than leaving the quality check to the end of the development. Reworking code early in the life cycle of program development costs less than rework later in the life cycle (6:1466). This is traceable to the cost of formal changes to approved systems and revalidation of the changed code (4:40). If a change results from an error early in the development cycle, no special approval milestones must take place. If, however, the error is discovered during the Maintenance phase of the development process, a full approval must occur with possible changes in the user manuals, and full retest and validation of the repair.

Other Goal Tradeoffs. Any particular programming goal can be traded off against any other. Boehm sites an experiment done by Weinberg in 1974 in which five programming teams were given the same programming assignments but each was told a different attribute to optimize. One team was asked to complete the program with the least effort (emphasis on schedule), another was asked to minimize memory required, another to minimize the number of statements, another was to produce the most understandable program, and the last was to produce the clearest output.

Each team finished first (or, in one case, second) in its area of emphasis but no team was able to do well on all objectives (4:20-21). This illustrates that different software goals conflict with each other and decisions should be made as to which to emphasize. By trying to emphasize an excessive number of attributes, a software developer will force the schedule and cost to increase.

Other goals may be traded off with cost if they are a higher priority, but modern software programming techniques may make that unnecessary. Using top down requirements analysis and code walkthroughs emphasizes the areas of the program which the programmers need to work on and allows a better check of the code correctness (4:676).

In top down requirements analysis a programmer progresses hierarchically from the top, most abstract level of the project to the more detailed levels. This prevents the designer from being bogged down in too much detail, allowing the designer to concentrate on one level at a time. The design is done iteratively; it is reviewed after each version and changes are made as needed (35:93). This process produces a more correct product from the start, minimizing later changes. Simpson further states that top down requirements analysis has been declared the best programming methodology for most programs (41:183).

A code walkthrough consists of a programmer guiding his supervisor or another programmer through the code produced to date. The combined

knowledge of the participants can find problems with the coding and the logic more easily than the programmer can alone. The walkthrough can also suggest alternate coding methods that are more efficient than that produced to date (35:15; 41:24-25).

Cost Estimating Methodologies

Software cost and schedule estimating techniques can generally be broken down into two broad categories: personal experience models and parametric models (32:5). A personal experience model relies on analogizing a proposed software project to a historic project of the same function and size. There are two main problems with this method. Many people must be queried, which is time consuming; and the project similarity, which is the basis for the estimate, is subjective (32:5). The second category, parametric models, is composed of regression, heuristic, and phenomenological models.

Regression. In a regression model a set of historic project costs is gathered, along with various parameters such as software size, number of input screens, etc., and a relationship is determined statistically (32:6; 38:4-2). This relationship is then used to estimate the cost of the proposed software project. This model enables the analyst to use confidence intervals, allowing the statistical risk to be quantified for the project. This method of cost estimation has the advantage of allowing size and cost estimation for projects which have no direct analogy (13:10-8). The regres-

sion, however, may not actually be a good predictor of software cost, especially if the data used in the regression is not indicative of the project the analyst is trying to estimate (24:62).

Heuristic. A heuristic model combines historic data with experience (32:6). Subclasses of this technique would be the data base analogy and Program Evaluation Review Technique (PERT) sizing (38:40-41).

The data base analogy is similar to the personal experience model except that a history of projects with their accompanying characteristics has been formally documented. The project to be estimated is broken down into its functional components and compared to similar projects from the data base. This method of estimating can be very accurate if the data base is current (38:40-41). Unfortunately, it is also possible to select a project which is not very similar, which will give erroneous results.

PERT sizing requires that experts in the software development field be consulted for cost estimates of any modules within the project. These estimates include the lowest, highest, and most likely costs. The three costs for each module are then weighted and averaged, producing an estimated module cost. This technique does not take into account unforeseen problems so it tends to be biased on the low side of the actual size experienced (38:38-39).

Phenomenological. Phenomenological models base software costs on a more general process. An example of this is estimating manpower

requirements using a Rayleigh Curve (32:6; 37:151-53). The Rayleigh Curve is a technique developed from diffusion theory (the theory studying the motion of molecules in a liquid environment). This process is used less often than others on an exclusive basis because the relationship between the model and the process is at times tenuous and the technique is difficult to understand. The Rayleigh curve in particular has been incorporated in many models that are mainly based on regression.

Sizing

The measurement used as the basis for many models' cost and schedule estimates is the size of the project (8:32). There are many measures available to estimate the size of a new project. One of these is to use a heuristic based on an expert's best guess of the lines of code needed for the project. A problem with this method is that not everyone can agree on what constitutes a line of code. The program may contain comment lines, blank lines, or non-executable lines, and a strict lines of code metric would count these. Most experts agree that a lines of code count should not include comments or blank lines but should include all the other types (8:34).

Another measure used to estimate program size avoids the problems of lines of code definition by estimating token count. Tokens are "basic syntactic units distinguishable by a compiler" (8:37). According to Conte, Halstead's "Software Science" states a token can generally be considered

either an operator or an operand (8:37). An operator would be a keyword or symbol that specifies an action, such as + or Add. An operand would be used to represent data within the action. This technique allows greater weighting of statements which have more executable parameters per line, as are allowed in some languages.

Yet another measure of size is the function count first proposed by A. J. Albrecht of the IBM Corporation in 1979 (42:68). The function point technique compensates for differences in lines of code sizing among languages by using parameters supplied by the estimator (number of inputs, number of outputs, number of inquiries, number of data files, and number of interfaces), weighting them by a formula developed by Albrecht, and adjusting the final functional estimate by a complexity factor for the project (42:70-72).

Software Development Models

This research was conducted with the aid of software development cost models that are commercially available or developed by the DoD. The cost models used in this analysis were determined by the availability of thorough documentation for the model. The following paragraphs describe the models considered for this research effort, with emphasis on the approach to schedule estimation.

PRICE-S. The PRICE-S model was originally developed by the RCA Corporation and is now owned and managed by the General Electric

Company. The principal inputs used by PRICE-S may be grouped into the seven categories shown in Table 2 (19:I-8).

Table 2
Primary PRICE-S Inputs

1) Project magnitude in SLOC
2) Program application (ie type of project)
3) Level of new design and code in SLOC
4) Productivity including experience and skill of programming team
5) Utilization of hardware (constraints)
6) Customer specifications and reliability requirements
7) Development environment

The basis for PRICE-S modeling is a "comparative evaluation of new requirements in light of analogous histories" (19:I-2). The PRICE-S model does not use the typical approach used by other models of applying a previously developed cost estimating relationship (CER). Instead the PRICE approach is more process oriented, utilizing the experience of the estimator to tune the results (19:I-13-I-14). By calibrating the data base for each user, PRICE-S, in effect, generates a new CER for each user.

PRICE-S examines any schedule restraints which are imposed and costs are adjusted for apparent stretch-out, acceleration, and phase transi-

tion efficiencies (19:I-2). As explained above, the schedule is "tuned" using the user's experience. This of course is a subjective assessment but the manual states that as much as possible "actual recorded data is used to formulate, test, and verify those assessment processes" (19:I-13). PRICE acknowledges that data is not always available. In this case the "impact of schedule variations on cost cannot be statistically processed" (19:I-13). However, the manual maintains that by knowing actual schedules differ from original planned schedules, cost impacts can be modeled by using the processes employed by the programming organization to manage the schedule (19:I-13-I-14).

Schedule outputs, as well as cost outputs, are made in matrix form, breaking out results by cost element and development phase (19:I-2). The classifications used are shown in Table 3 (19:I-1).

PRICE-S also examines Software Operations and Support. Outputs for this area include maintenance, enhancement, and growth (19:I-14). Once again, the output is generated by Cost Element.

Ray's Enhanced Version of Intermediate COCOMO (REVIC).

REVIC is a model developed by Raymond L. Kile of the Air Force Contract Management Division, based on the intermediate COnstructive COst MOdel (COCOMO) described by Dr. Barry Boehm in his book Software Engineering Economics (4). COCOMO itself is one of the more popular models used because Boehm published all the equations for his

Table 3
PRICE-S Outputs

Cost Elements	Development Phases
Design	System Concept
Programming	System/Software Requirements
Data	Software Requirements
System Engineering/ Program Management	Preliminary Design
	Detail Design
Quality Assurance	CSC Code/Unit Test
Configuration Management	System Test and Evaluation
	Operational Test and Evaluation

model and there is no fee for usage.

The differences between REVIC and the intermediate COCOMO defined by Boehm are that REVIC adds some parameters not used by Boehm (security-SECU, management reserve-MRES, and required reusability-RUSE), and that the coefficients used by REVIC are somewhat different than those used by Boehm because the model was calibrated using a DoD data base (29:4). REVIC also includes an additional development period called system engineering which takes place before preliminary design, and a period called development test and evaluation which takes place after integration and test (29:2). These two additional periods will be considered during normalization.

The COCOMO equations were obtained by analyzing a data base of 63 software projects from mixed sectors of the software world (4:83). Boehm developed three different versions of COCOMO. One, Basic COCOMO, is the most simple to use and most useful in the early stages of the life cycle when little is known about the project and rough estimates are sufficient (4:58). Intermediate COCOMO incorporates an additional 16 predictor variables to more accurately estimate cost and schedule (4:114). Detailed COCOMO uses the equations from Intermediate COCOMO and adds a three level hierarchy to estimate software development projects from the ground up (4:348).

Boehm developed his 16 variables from reading various studies and reducing the number of candidates using two main tests:

1) General Significance--eliminated factors which were significant only a small fraction of the time (4:115).

2) Independence--eliminated factors which were strongly correlated with product size and compressed a number of factors which tend to be correlated to a single factor (4:115).

The resulting factors in Intermediate COCOMO are broken into four general categories: software product attributes, computer attributes, personnel attributes, and project attributes (4:115). The factors for each category are listed in Table 4.

The Intermediate COCOMO software development model begins by

Table 4
COCOMO Factors by Category

Product Attributes
Software Reliability (RELY)
Data Base Size (DATA)
Product Complexity (CPLX)
Requirements Volatility (RVOL)
*Required Reusability (RUSE)
Computer Attributes
Execution Time Constraint (TIME)
Main Storage Constraint (STOR)
Virtual Machine Volatility (VIRT)
Computer Turnaround Time (TURN)
Personnel Attributes
Analyst Capability (ACAP)
Applications Experience (AEXP)
Programmer Capability (PCAP)
Virtual Machine Experience (VEXP)
Programming Language Experience (LEXP)
Project Attributes
Modern Programming Practices (MODP)
Use of Software Tools (TOOL)
Required Development Schedule (SCED)
*Management Reserve (MRES)
*Security Requirements (SECU)

Note: Requirements Volatility was added in 1986

* denotes factors used in REVIC but not in Intermediate COCOMO

generating a nominal effort estimate using the following equations:

$$1) MM = A\,(KDSI)^{B} \prod F_i$$
$$2) TDEV = C\,(MM)^{D}$$

Equation 1 predicts the manpower (MM) in man-months based on the estimated lines of code to be delivered (KDSI is an acronym for Thousand Delivered Source Instructions) and the product of environmental factors F_i. The Coefficients (A, B, C, and D) and the factors (F_i) are determined by statistical means from a data base of completed projects. . . . The results from equation 1 are put into equation 2 to determine the resulting schedule (TDEV is an acronym for Time for Development) in months needed to perform the complete development (29:1).

The output of Intermediate COCOMO is the level of effort in person-months (4:115). A COCOMO person-month consists of 152 hours of working time which takes into account time off due to holidays, vacation, and sick time (4:59). COCOMO estimates in person-months rather than dollars because of "the large variations between organizations in what is included in labor costs . . ." (4:61).

In the sphere of schedule COCOMO forecasts the "average labor level estimates for each software development phase . . . " using the Rayleigh Distribution (4:67). Boehm found the Rayleigh curve to be a "reasonably good fit for portions of the manpower distribution . . . with the main exception of its zero-level behavior at the start of the project" (4:93). Thus, COCOMO uses a modified form of the Rayleigh curve as its basis for schedule estimation.

SASET. The Software Architecture Sizing, and Estimation Tool is a model developed for the Naval Center for Cost Analysis by Martin Marietta Denver Aerospace Corp. and is a "forward chaining, rule-based expert system utilizing a hierarchically structured knowledge database of normalized parameters to provide derived software sizing values by functionality, an "optimal" software development schedule, and associated manloading charts (31:I-2)". This system combines the best features of bottom's-up estimating and parametric methodology (31:I-2).

SASET wields a three tiered approach to system identification with a fourth tier dedicated to maintenance and a fifth tier to risk analysis (31:I-2). The first tier addresses the class of software, the programming language, the development schedule, and other environmental issues (31:I-2). This data is used to generate a Software Budget Multiplier and a Software Schedule Multiplier (31:III-1).

The second tier specifies the functional aspects of the system which then are used to forecast the size of the software (31:I-2). The functional inputs include whether the software is generated from scratch or modified from existing sources, and the language used (31:I-2). An alternative is to enter the size of the system directly which is then used to generate preliminary budget and schedule forecasts (31:I-3).

The third tier describes the software complexity issues of the hardware/software system including system timing and criticality, and

documentation required (31:I-3). The inputs from this tier are used to generate a Software System Budget Multiplier and Software System Schedule Multiplier which are used to generate the final budget and schedule predictions (31:III-11).

The fourth tier addresses the maintenance life cycle and requires entry of 17 complexity factors (31:I-3). The fifth tier provides risk analysis on sizing, schedule, and budget data through the use of user entered distribution parameters and Monte Carlo simulation (31:I-3).

The final schedule that is generated is based on the size of the software available in tier two and an average of the Software Schedule Multiplier from tier one and the Software System Schedule Multiplier from tier three (31:III-17). Final schedule estimates span from a Systems Planning Review to an Acceptance Review (31:VI-13).

Output reports include sizing estimates, schedule estimates, man-loading estimates, effort estimates, budget and schedule factors, tier input review, maintenance and risk assessment (31:iv).

SEER. System Evaluation and Estimation of Resources is a model published by Galorath Associates, Inc. and is based on the work of Randall Jensen (17:V-1). Dr. Jensen first published his software estimation model in a paper titled "An Improved Macrolevel Software Development Resource Estimation Model". Using a technology constant based on technolo-

gy input parameters and the Rayleigh-Norden curve, SEER computes the required software development effort in staff-months and dollars (28:1).

SEER uses four primary inputs plus development constraints to generate the schedule and effort for Software Requirements Analysis through CSCI Integration and Testing (17:V-8). The primary inputs are effective size to include both new code and software to be modified, effective technology rating which accounts for variations observed in schedule and effort for similar projects, complexity which encompasses differences in operating systems and interfaces, and the staffing rate (17:V-8-V-9).

The primary inputs for SEER are presented in a Microsoft Windows environment and each input requires three values; one for lowest likely value, one for most likely value, and one for the highest likely value (17:X-1). Galorath and Associates purports that this method of data entry permits the users to bound the risks and uncertainty in the parameter itself (17:X-1). Inputs for SEER fall under the following parameter categories: size or source lines of code (SLOC), personnel capabilities, development support environment, product development requirements, product reusability requirements, development environment complexity, target environment, and schedule (17:Appendix E).

Some of the key reports produced by SEER include: a Quick Estimate Report, a Basic Estimate Report, a Maintenance/Operation

Support Report, an Inputs Report, a Staffing by Month Report, a Cost by Month Report, and a Cost by Activity Report (17:XI-1-XI-2). Reports regarding the projects risk include a Risk by Cost Report, a Risk by Person Months Report, and a Risk by Schedule Report (17:XI-2).

SEER contains a Quick Estimate Window which is on screen at all times. This window contains the estimates for schedule months, effort months, base year cost, effective size, and the life cycle phases included in the estimate (17:III-3). The window will show the effect that each parameter has on the estimates as they are input by updating the estimates after data entry.

The schedules generated by SEER include software requirements analysis, software design through CSCI test complete, software integration through operational test and evaluation, and software maintenance/-operational support (17:XI-6). The on screen Quick Estimate may be easily modified to include only the portions of the schedule required by using the Estimate/Quick Estimate Options from the menu (17:VIII-9).

SLIM. Software Life Cycle Model (SLIM), a model developed by Quantitative Software Management, Inc., is based on the work of Putnam (36:xx). The algorithm's used in SLIM require the use of descriptive information about the system being developed and the organization's capabilities to code the effort (36:III-4). The system information required includes the availability of target computer memory, the application type

being coded, and the languages used (36:III-4). The developer information includes personnel experience, software engineering techniques used, and the general type of computer used in development (36:III-4).

The model may also employ two index numbers: a Productivity Index and a Manpower Buildup Index (36:III-4). The Productivity Index calibrates the model to the efficiencies of the developer while the Manpower Buildup Index determines the maximum effective staffing rate for a project (36:III-4). These indices may be calculated from a developer's historical data or can be approximated by SLIM using industry averages (36:III-5).

The final input to SLIM is the size of the system to be developed in SLOC (36:III-5). As in SEER, the model requires the smallest likely, the most likely, and the largest likely values for this parameter (36:III-5).

The schedule generated by SLIM is a minimum time solution based on a Monte Carlo simulation (36:III-6). Using a longer schedule than that generated by SLIM will reduce the cost of the system (36:III-7). The costs required will be calculated by SLIM if the required constraint, such as maximum staffing, maximum cost, or acceptable reliability level, is input (36:III-7).

SLIM reports include Life Cycle Reports covering schedule, effort, staffing levels, milestones, and cashflow, Risk Analysis Reports covering probability profiles for meeting schedule and effort, a Work Breakdown

Plan, a Code Production Plan, a CPU Usage Plan, a Reliability Plan forecasting defects, and a Documentation Plan estimating the number of pages of documentation (36:VIII-1-VIII-67). The model also prints charts displaying the above results, as well as a Gantt Chart for managing the project (36:VIII-1-VIII-67).

SPQR/20. This model is marketed by Software Productivity Research, Inc., and is based on the work of Capers Jones. The acronym, "SPQR", stands for software productivity, quality, and reliability; the primary outputs of the model. The "20" refers to twenty multiple-choice and other questions which must be answered by the user for model inputs which describe the user's experience, development methods, and environment (42:2).

Features of the model include the number of source code statements required for a project or the number of function points, the development schedules by phase, the development effort and the staff sizes by activity (42:2). SPQR/20 also predicts the number of defects which can be expected in the project, the number that can be removed by various tests, and the number which can be expected to remain in the product on delivery (42:2). SPQR/20 also predicts the quantity of documentation required to support the developed system (42:2).

The major included and excluded activities for SPQR/20 are shown in Table 5 (42:2).

Table 5
SPQR/20 Activities

Included by SPQR/20	Excluded by SPQR/20
Planning	User Effort
Requirements	User Education
Design	Quality Assurance
Coding	Hiring and Moving
Integration	Production Costs
Testing	Field Service
Documentation	Customer "Hot Lines"
Management	Travel Expenses
Central Maintenance	Capital Equipment
Enhancements	Staff Education

The developers of SPQR/20 foresaw that one or more of the above activities that are excluded by the model could be a cost driver. Software Productivity Research, Inc. provides an input called "Other Costs" which may be used to pass the costs estimated for the excluded activities to the project cost (42:3).

SPQR/20 can provide an estimate of schedule based on thousands of lines of code (KLOC) or can provide an estimate of schedule and lines

of code based on function points (42:37). (The theory of function points has been discussed earlier.)

According to the SPQR/20 manual, the advantages of the function point technique are that 1) Function points are independent of source code and do not penalize higher order languages, 2) function points can be applied early in the program life cycle, and 3) function points can be used to predict source code size (42:69). The disadvantages listed are 1) there is substantial ambiguity in the exact definition of function point parameters and 2) the treatment of complexity was purely subjective using a range of ±25% (42:70).

SPQR/20 outputs include risk and quality estimates, defect removal and reliability estimates, the main development and maintenance cost estimates, and normalized managerial information (42:46).

System-4. System-4 is a software cost estimating model developed by Computer Economics, Inc. (CEI) and is based on the work of Randall Jensen (7:I-4). Just as in SEER , System-4 uses a technology constant based on technology input parameters and the Rayleigh-Norden curve, to compute the required software development effort in staff-months and dollars (28:1).

CEI first developed a model for software estimation in 1979. JS-1 and JS-2 preceded System-3 which in turn was followed by System-4 (7:I-4).

The primary inputs for System-4 require three values; one for lowest likely value, one for most likely value, and one for the highest likely value (7:IV-4). Inputs for System-4 fall under the following parameter categories: size or source lines of code (SLOC), developer personnel, developer environment, target computer environment, product development environment, development support environment, and program schedule (7:Input Sheet).

Some of the key reports produced by System-4 include: an Input Parameter Report, a Summary Report which includes schedule and cost, a Cost Spread Report which provides detailed cost breakdowns by task phase and labor category, a Staffing Report, and a Risk Analysis Report (7:VIII-2-VIII-7).

System-4 contains a Solution Window which is on screen at all times. This window contains the estimates for Design Implementation and Test (DIT) schedule, Full Scale Development (FSD) schedule, FSD cost, peak staff, lines of coding per month, technology rating and effective size of the project (7:IV-5). The window will show the effect that each parameter has on the estimates as they are input by updating the estimates after data entry.

The schedule generated by System-4 includes software requirements, architectural design, detailed design, code and unit test, integration and test, and system integration and test (7:VII-28). The on screen Solution

Window carries the total times for Design Implementation and Test (which includes architectural design, detailed design, code and unit test, and CSCI integration and Test) and Full Scale Development (which includes the portions scheduled in DIT as well as software requirements and system integration and test (7:VII-28).

Previous Efforts

There have been a handful of early efforts to evaluate the accuracy of model schedule estimates. Among these has been a study by Illinois Institute of Technology Research Institute (IITRI), an evaluation by MacDonald Dettwiler and Associates (31) (both including schedule among other measures to rate), an AFIT thesis by Captain Crystal Blalock (2), and a report for the US Army Aviation Systems Command (23).

IIT Research Institute. The study by IIT Research Institute compared the ability of six models, Ada COCOMO, Softcost Ada, PRICE-S, System-3, SPQR/20, and SASET, to estimate the cost and schedule of Ada projects (26:vii). IITRI used a data base of eight completed Ada projects from various sources to test the models. The results were broken down into six categories: 1) overall effort, 2) overall schedule, 3) government contracts, 4) commercial contracts, 5) command and control applications, and 6) tools and environment applications (26:vii-viii).

The results indicate that System-3 and Price S are the most accurate models for schedule estimation for the data base studied. Estimates using

System-3 were within 30% of actuals on four of the eight projects while estimates using Price S were within 30% on three of the eight (26:ix). The test results, however, can not be generalized because all of the projects were Ada projects and the data base was very limited.

MacDonald Dettwiler and Associates. The evaluation performed by Rick Martin at MacDonald Dettwiler and Associates tested the Before You Leap Model (based on COCOMO), WICOMO (based on COCOMO), System-3, and SPQR/20 (31:49). Martin used a data base of seven completed projects from the MacDonald Dettwiler company to compare the models. The models were ranked in the areas of user interface, input data required, output data meaning, and accuracy of cost and schedule predictions (31:49).

Results for the models in this study were tabulated in a slightly different manner than the IITRI study. The estimates were compared to the actual schedules using the method of Least Squares Best Fit, with the actual schedule being treated as the independent variable (31:50). The results showed the Before You Leap Model of COCOMO to be the most accurate (lowest Standard Error of the Estimate) followed by SPQR/20 (31:50). As a comparison for the IITRI study, all errors for the Before You Leap Model were within 30%, and all but one of the seven were within 30% for SPQR/20.

This analysis has the same weak point pointed out above in that no inference may be made on the basis of a test using seven data points. Additionally, the test cases were all accomplished at the same company, indicating the accuracy of the models may not be implied to other agencies.

Capt. Blalock. An AFIT thesis accomplished by Capt. Crystal Blalock was directed specifically at determining the accuracy of models at estimating schedule for software development projects. Blalock, as a potion of her thesis, tested COCOMO, PRICE-S, System-3, SoftCost-R, and SPQR/20 (2:37). The study used one data point representing the average project from a 31 point data base.

The results showed that all of the models except COCOMO were reasonably competent at estimating the data point. All of the models estimates were within 20% except the COCOMO model used (calibrated for Electronic Systems Division). System-3 was the most accurate (2:87).

This study exhibits the same flaw mentioned above. It was limited in its scope to a small data base (in this case extremely small). Because the study used only one point it also did not use any statistical analysis of results.

US Army Aviation Systems Command. Richard M. Greathouse and Kelly L. Shipley published a report for the Directorate for Systems and Cost Analysis at the US Army Aviation Systems Command comparing the schedule estimating accuracy of four models. The four models chosen were

SECOMO (an adaptation of Boehm's COCOMO), REVIC, PRICE-S, and SASET (23:1-2). The study used 12 data points to compare the models (23:3).

The study concluded that SASET was the most accurate estimator of schedule, while SECOMO underestimated and PRICE-S and REVIC overestimated the required schedule (23:4). The study further found that the three less accurate models drastically overestimated the Preliminary Design phase of the software development life cycle (23:4). When this error was removed PRICE-S and REVIC had less variance in the estimate than SASET, the best estimator (23:5).

This study is the most convincing one to date. The data base consisted of 12 data points which strengthens the study. Also the evaluators used some statistical analysis with which to judge the models. The analysts, however, used the average of the resulting schedules to evaluate the accuracy of the models. This is not necessarily the best way to judge the precision.

III. Methodology

To discover what available programs best estimate software development schedule requires modeling historical data. The first step to be taken will be to obtain information concerning what software development schedule estimators are available to the Air Force. A source of this information was the faculty at the Air Force Institute of Technology and analysts at Electronic Systems Division and Aeronautical Systems Division. After locating these, it was necessary to find out which may be purchased or borrowed by AFIT for evaluation (this special purpose software can be very expensive).

The selection of models was based on the availability of documentation regarding the schedule generation conducted by the model and the availability of the model itself. Models considered for use were:

1) PRICE-S
2) Ray's Enhanced Version of Intermediate COCOMO (REVIC)
3) SASET
4) SEER
5) SLIM
6) SPQR/20
7) System-4

Concurrent with the investigation for models to test was an Air Force Wide search for an available data base of accurate measurements from authentic software engineering efforts.

Once these two objectives were met, the information from the data base was interpreted for each of the selected models. This was necessary because each model uses its own set of environmental factors as input which aren't necessarily the same from model to model. Since Professor Ferens is a recognized expert in this field, his help was solicited.

The data was then run in each model to generate an estimate of the development schedule. For those categories of input for which no data was available within the data base, the default values supplied by the model, if any, were used. For those instances in which the data base had an appropriate category but the data was missing, a nominal rating was assumed for that parameter. The PRICE-S value used for PROFAC (Productivity Factor) was 4.00. The default parameter sets used in the models were predominantly command and control, and radar, to match the types of projects being estimated.

Once this estimate had been obtained for each set of data, the results were normalized to match the portions of the life cycle reflected in the data base (in this case, the portions of the life cycle used in COCOMO). Normalization was needed because the models each forecasted

different sections of the software life cycle. The estimates were then compared to the actual schedule for each project.

Tests

Various statistical tests were used, both parametric and non-parametric, to evaluate which of the models was most accurate at projecting the development schedule. They include Least Squares Best Fit (LSBF) using a linear relation (Actual = β_1*Estimate + β_0), Least Squares Best Fit using a log-log relation (Actual = (β_0*Estimate)^β_1), the Wilcoxon Test, the Friedman Rank Sum Test, and the percentage method used by various developers to evaluate models (within X% of the Actuals Y% of the time). Levels of uncertainty (α s) of .05 and .01 were used for all tests unless otherwise specified.

Linear Least Squares Best Fit. The statistical method of Least Squares Best Fit was used to discover which model is best at estimating the required software engineering schedule. The seven basic assumptions which must be met in order to use the method of Least Squares are listed in Table 6.

A hypothesis test was performed on the significance of the relationship for each model. To execute this test the F Ratio is used. The F Ratio is a measure of the relationship between x and y. If this is large, it means that x and y seem to have a significant relationship. The F Ratio is used to test the hypothesis of whether the estimated schedule is related to

Table 6
Least Squares Assumptions

1) The εs (error terms) of the relation are normally distributed.

2) The εs have a variance of σ^2, in which σ^2 is a constant.

3) The expected value of the εs equals 0.

4) The εs are independent (not correlated with each other).

5) The sample data selected are truly representative of the population.

6) The sample data are random.

7) The independent variables are independent of each other.

the actual schedule at a significant level. We may state this in this way:

$$H_o : \beta_1 = 0$$

$$H_a : \beta_1 \neq 0$$

in which β_1 is the slope of the estimating equation. If the null hypothesis cannot be rejected, this means that this data does not support the existence of a significant relationship between the actual schedule and the estimated schedule.

The comparison among the models was done on the basis of the amount of variation explained by all the statistically significant models. This may be measured by the Coefficient of Determination (also known as

R^2). This is defined as the proportion of the total variation that is explained by the model (the Sum of Squares of the Residuals divided by the Sum of Squares of the Total, or SSR/SST).

Log-Log Least Squares Best Fit. There is some question whether the error terms have a constant variance over the range of the population. Some experts believe that the error terms increase as the estimate gets larger (16). This problem may be solved by performing a LSBF on the log of the Estimates versus the log of the Actuals. A hypothesis test was used as described above, and the Coefficient of Determination was used to judge the strength of the relationship.

Wilcoxon Sign Test. The Wilcoxon Sign Test is used to investigate for bias. The test requires finding the differences of the estimate schedules and the actual schedules, and ranking the differences. The rankings are then divided up into positive and negative columns and the ranks of the columns are summed. The smaller of the two results is called the Wilcoxon T statistic. The standard normal distribution is then used to test whether the mean of the differences is zero (there is no bias) versus the mean is not zero (there is bias).

If the T statistic is large enough, it suggests that the rankings of the positive and negative differences are more or less equal. This would be expected if there were no bias in the model. A biased model would create estimates which are usually high or low, creating a majority of ranks in one

column or the other, generating a Wilcoxon T statistic that is low (the lower rank is used as the value) (34:398-401).

Friedman Rank Sum Test. The Friedman Rank Sum Test may be used in two different ways. In both methods the actuals and the estimates are placed in a table and each observation is considered individually. The estimates and the actual value for each observation are ranked in size with the smallest being one. The ranks for each model and the actuals column are summed across all observations.

In the first use of these figures, the rank sums are used to calculate an S statistic which is computed as follows:

$$S=\left[\frac{12}{nk(k+1)}\sum_{j=1}^{k}R_j^2\right]-3n(k+1)$$

in which n = the number of observations in the columns

k = the number of columns

and R = the ranks of the various columns

The S statistic is then tested versus a χ^2 statistic for a given level of uncertainty (α) for k-1 degrees of freedom. If S_{calc} is greater than the χ^2 value, we reject the null hypothesis which is that all of the data sets (actuals and all model estimates) are the same.

The S statistic is basically a test of whether or not all of the ranks are approximately equal. If this were the case it could be inferred that the

rankings of the model's estimates and the actual values were purely random and that all columns were relatively equal. If the rankings differ significantly, it may be concluded that there is a disparity between the column's values (25:139-141).

In the second use of these figures, the rank sums of the estimates are compared to the rank sum for the actuals column to detect if there is a significant difference between each rank (each set of estimates versus the actuals). Each of the differences (actual rank sum versus estimate rank sum) is compared to an m^* statistic derived using an m statistic from a table. The equation is as follows:

$$m* - m(\alpha, k-1, \frac{1}{2}) \left[\frac{nk(k+1)}{6} \right]^{\frac{1}{2}}$$

in which n = the number of observations in each column

and k = the number of columns

If the difference is greater than the computed m^* value, the null hypothesis (that both sums are equal) is rejected.

The m^* statistic is similar to the S statistic discussed above. If the value is small this implies that the rankings are approximately equal and there is no difference between the two columns. If the ranks diverge profoundly, the test will fail and it may be deduced that the two columns are not roughly the same (25:155-157).

Percentage Method. The estimated schedules are divided into the actual schedules and are compared to a predefined percentage. The models are then compared to see how often each has estimated within the predefined percentage. This is the method used by many model developers to tout their products because this method is the easiest to use and to understand.

IV. Findings

Model Selection

Five models were selected for this study. Several models were considered but were rejected for a number of reasons. SASET was not chosen because of the time involved in learning and running the model. The evaluator did not feel comfortable with the amount of time available to learn this complex model. Further, the model is not available outside the Department of Defense and it was hoped that the results of this thesis would be of broader application.

The SLIM model was not chosen because that model is no longer available at the Air Force Institute of Technology.

Data Base Selection

First, the Ballistic Missile Office of Norton AFB was contacted to investigate as to whether the data base that was used to develop the Ballistic Missile Office Software Cost Model was available. However, the data used to develop that model is not Air Force data and is the same data used by Dr. Boehm to develop his COCOMO model. The Electronic Systems Division of Air Force Systems Command was then contacted to examine their software cost data base. However, the data maintained by the Electronic Systems Division, while beneficial, was not of sufficient detail to run the models required in this thesis. The MITRE corporation

was then contacted to inquire about their data base developed from Electronic Systems Division programs. This data base was chosen because of its availability and completeness of parameters.

Data Base

The data base consisted of 26 projects performed by various contractors for the Electronic Systems Division of Air Force Systems Command. Of the 26 projects, 21 data points contained the actual schedules experienced during development. The size of the projects ranged from nine thousand to over one million lines of code.

The projects' mission descriptions were primarily command and control systems and radar systems. There were a few simulation and training systems. Schedules ranged from 13 to 84 months. Unfortunately, the data are proprietary and could not be reported with the results. Because of this, the output runs were also not included.

Linear Least Squares Best Fit

The models all generate significant F values at both 95% and 99% levels of confidence, which indicate that in all cases, the model is a better estimator than the historical averages. However, the fits produced are not very good. The coefficients of determination (R^2) ranged from approximately 0.31 to 0.43, meaning less than half of the total variation is being explained by the regression line. Results are shown in Appendix A.

Log-Log Least Squares Best Fit

Just as in the linear least squares best fit, all of the estimators were significant at the 95% and 99% levels of confidence, yet the actual fit produced by the estimators was not good. In the case of a log-log least squares, the coefficients of determination ranged from 0.34 to 0.49. Once again, none of the estimators explained more than 50% of the total variation from the actuals. Results are shown in Appendix B.

Wilcoxon Test

The Wilcoxon Test indicated that there was bias in all model estimates except for that of System-4 at 95% and 99% level of confidence, and PRICE-S only at 99% level of confidence. Detailed results are in Appendix C.

Friedman Rank Sum Test

The Friedman Rank Sum Test was first used to identify whether or not all of the estimates and the actuals were approximately equal. This would be indicated by all of the ranks being about the same (no one data set would be predominately high or low). The test indicates that this is not the case. The S statistic that was calculated was 47, while the χ^2 statistic against which to test the S was 11 at 95% confidence and 15 at 99% confidence. This demonstrates that there are differences between the columns. This test gives no indication of which are different.

The second use of the Friedman Rank Sum Test compares the rank sum of the data set of interest (in this case, the set of actual schedules) against all of the rank sums of the other data sets. The difference is then compared to an m^* statistic. The data set from System-4 easily passes the test while PRICE-S barely passes the test at the 95% level (PRICE-S would not pass at any smaller confidence level). REVIC passes only at the 99% level. This means that the data in the System-4 column and the Actuals column can be considered about equal, as can PRICE-S and the actuals, and REVIC and the actuals at the 99% level of confidence (see Appendix D).

Percentage Method

The percentage method was administered twice (see Appendix E). Initially the percentage used to compare the models was 30%. Using this band around the actuals, System-4 showed itself to be within 30% of the actuals 71% of the time. The next closest estimators were REVIC and SEER, being within 30% of the actuals 33% of the time.

The analyst wished to find out how well the estimators forecasted a little closer to the actuals so the percentage was lowered to 20%. When this was done System-4 and SEER tied for the best estimators at 28.6%. This indicates that within this 10% band (from 20% to 30%), SEER only captured one more observation while System-4 captured nine more).

A common method used by price analysts in evaluating a contract proposal is to apply a decrement factor to the proposal for any known overestimation. Since all the models but System-4 were known to overestimate (Appendix C), this method was applied to the results of the models to observe whether the models could be adjusted to produce a better estimate band than System-4.

The models were evaluated to detect the average percentage that was overestimated. The inverse of this number was then multiplied by all of the ratios of estimates to actuals. As can be seen in Appendix F, when this process was used all of the models forecasted better within the 20% band (REVIC, PRICE-S, and System-4 were significantly better). Within the 30% band, all of the models forecasted closer to the actuals except for System-4, which got worse. This is due to the inverse multiplier being affected by extreme values, forcing the borderline cases (project numbers two and four for example) outside the band. In no case, however, were any of the models more accurate than System-4 within the 30% band.

It was then postulated that a similar situation regarding the influence of extreme data points may occur with the other models. There might be different percentages which could be used to adjust the other models which would generate better estimates than those shown with the inverse. In this case the analysis was held to a 30% band because the 20% band

required more adjustments than would normally be used and a model was already good at estimating within the 30% band.

The percentage decremented was varied in 10% increments for all models except System-4, which was used as a baseline. The models all approached the accuracy of System-4 until they peaked using a 40-50% decrement. At that point all of the models started to further deviate from the actuals again. Once again, no model forecasted as closely to the actuals as System-4.

It was then conjectured that perhaps one segment of the software development life cycle estimates was affecting the accuracy of the models. The average deviation from the actuals by life cycle segment by model is

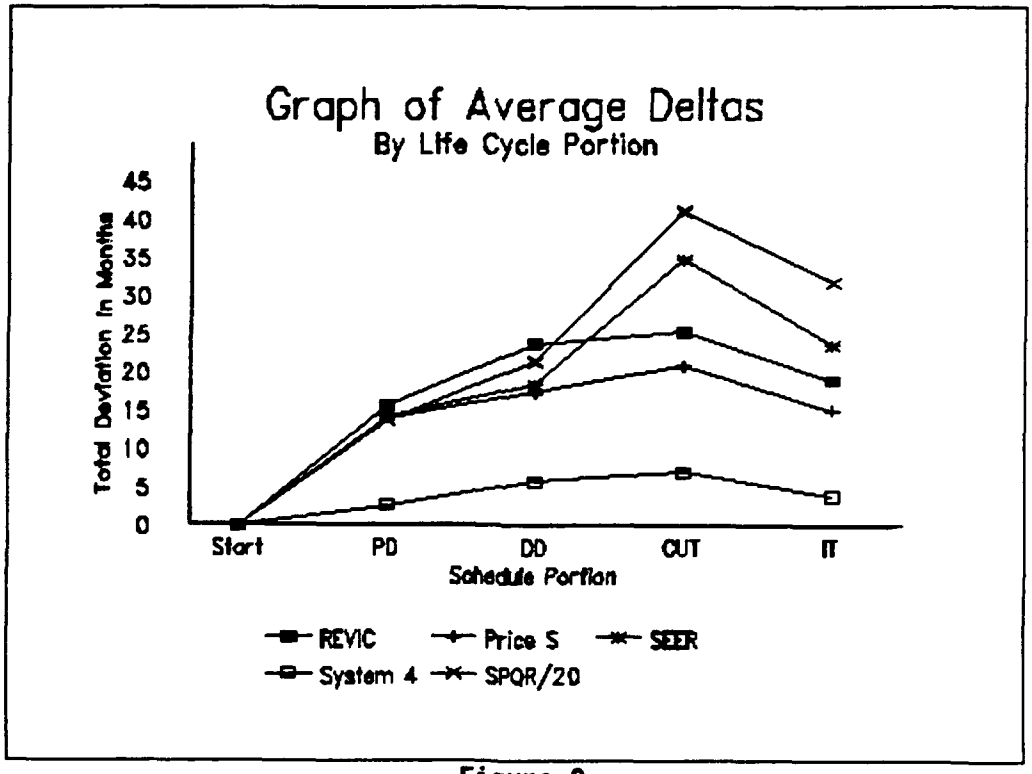


Figure 2
Life Cycle Estimate Comparison

shown in Figure 2. The acronyms used are explained as follows:

PD stands for Preliminary Design

DD stands for Detailed Design

CUT stands for Code and Unit Test

and IT stands for Integration and Test.

A model which produced no deviation from the actuals would be a straight line along the X axis. System-4, which emerges as the most

accurate model by all of the non-parametric tests, produces a line closest to this ideal. It is evident that if the deviation from the actual schedules during the Preliminary Design portion of the life cycle could be removed from REVIC and PRICE-S, they would be roughly as accurate as System-4.

V. Conclusions and Recommendations

Conclusions

Much research has been done in the field of software development but little in the realm of schedule accuracy. A few studies have been performed in this area but all were flawed on various grounds. For this reason five software development models were evaluated using a data base developed by MITRE for the Electronic System Division (ESD) of Air Force Systems Command. The five models chosen were REVIC, PRICE-S, SEER, System-4, and SPQR/20. Using the 21 data points from the ESD data base which had a schedule identified, the models were run and data evaluated both parametrically and non-parametrically. While no model was shown to be a better predictor than any other on the basis of parametric measures, System-4 proved overwhelmingly to be the best using non-parametric tests. It was also found that REVIC and PRICE-S could essentially match System-4 if the overestimation in the Preliminary Design phase of the software development life cycle could be negated through calibration or some other mode. In PRICE-S this is possible through the use of schedule phase multipliers. REVIC has not yet implemented a method to adjust the schedule phase distributions.

Recommendations

Further research in the area of schedule determination for software development programs is needed in order to help estimators more accurately calculate these schedules. There are other models available which were not used because of various reasons. The Electronic System Division data base should be run in those models to test their accuracy against the results encountered in this study.

It would also be beneficial to locate another data base to test the models used. The results from this study can only be applied to similar programs to those in the data base used, command and control and radar systems. A different data base would allow a more general application of results.

Finally, it would be advantageous to test the models using the Electronic System Division data base in the method used for this study, but with the analyst calibrating the models with a portion of the data base. This method would test the models as a true expert would use them, in a mode calibrated to calculate the types of projects being estimated.

Appendix A: LSBF Analysis

Input Data

Project #	REVIC	SPQR/20	System 4	SEER	PRICE-S	Actuals
1	32.3	36.9	28.5	31.6	45.7	29.0
2	19.7	35.7	19.0	22.2	22.8	27.0
3	28.0	29.7	23.0	30.8	30.4	18.0
4	25.6	28.2	24.1	30.3	36.9	33.0
5	25.9	39.9	13.5	26.8	29.3	13.0
8	61.1	86.1	44.6	66.7	48.0	68.0
9	62.0	109.9	44.9	62.7	43.9	71.0
10	141.9	116.1	107.3	158.0	131.6	84.0
11	51.9	73.0	30.4	55.3	39.2	22.0
12	61.8	77.1	45.9	68.0	53.2	75.0
13	71.1	103.1	52.6	62.5	62.7	43.0
14	54.7	83.7	55.2	54.1	42.6	23.0
15	143.3	83.4	99.9	165.2	114.9	41.0
17	42.8	68.9	27.0	45.7	44.1	22.0
18	52.8	69.9	34.6	56.7	48.8	31.0
19	53.5	74.2	30.0	66.7	50.0	40.0
20	40.3	84.0	26.1	48.7	54.9	26.0
21	42.3	64.5	30.9	42.6	34.2	33.0
22	46.1	70.9	31.4	50.6	44.8	25.0
23	56.3	71.6	25.4	69.2	55.1	33.0
24	39.5	30.9	23.2	46.1	36.3	26.0

LSBF Results

REVIC : Regression Output:

Constant	16.67358		
Std Err of Y Est	16.40234	SSE =	5111.696
R Squared	0.364714	SST =	8046.286
No. of Observations	21	SSR =	2934.59
Degrees of Freedom	19		
		F =	10.90777
X Coefficient(s)	0.375546		
Std Err of Coef.	0.113709	F(.95,1,19)	4.38
		F(.99,1,19)	8.19

SPQR/20 : Regression Output:

Constant	3.031709		
Std Err of Y Est	15.55677	SSE =	4598.251
R Squared	0.428525	SST =	8046.286
No. of Observations	21	SSR =	3448.035
Degrees of Freedom	19		
		F =	14.2473
X Coefficient(s)	0.50034		
Std Err of Coef.	0.132556	F(.95,1,19)	4.38
		F(.99,1,19)	8.19

System 4: Regression Output:

Constant	17.17003		
Std Err of Y Est	16.12817	SSE =	4942.24
R Squared	0.385774	SST =	8046.286
No. of Observations	21	SSR =	3104.045
Degrees of Freedom	19		
		F =	11.93322
X Coefficient(s)	0.51686		
Std Err of Coef.	0.149621	F(.95,1,19)	4.38
		F(.99,1,19)	8.19

SEER : Regression Output:

Constant	18.26531		
Std Err of Y Est	16.76138	SSE =	5337.936
R Squared	0.336596	SST =	8046.286
No. of Observations	21	SSR =	2708.35
Degrees of Freedom	19		
		F =	9.640177
X Coefficient(s)	0.357162		
Std Err of Coef.	0.115033	F(.95,1,19)	4.38
		F(.99,1,19)	8.19

Price-S : Regression Output:

Constant	15.42216		
Std Err of Y Est	17.09459	SSE =	5552.277
R Squared	0.309958	SST =	8046.286
No. of Observations	21	SSR =	2494.009
Degrees of Freedom	19		
		F =	8.534546
X Coefficient(s)	0.429338		
Std Err of Coef.	0.146964	F(.95,1,19)	4.38
		F(.99,1,19)	8.19

Appendix B: Log-Log Analysis

Project #	REVIC	SPQR/20	System 4	SEER	PRICE-S	Actuals
1	1.509	1.567	1.455	1.499	1.660	1.462
2	1.294	1.553	1.279	1.347	1.358	1.431
3	1.447	1.472	1.362	1.488	1.483	1.255
4	1.408	1.451	1.382	1.481	1.567	1.519
5	1.413	1.601	1.130	1.428	1.467	1.114
8	1.786	1.935	1.649	1.824	1.681	1.833
9	1.792	2.041	1.652	1.797	1.642	1.851
10	2.152	2.065	2.031	2.199	2.119	1.924
11	1.715	1.863	1.483	1.742	1.593	1.342
12	1.791	1.887	1.662	1.832	1.726	1.875
13	1.852	2.013	1.721	1.796	1.797	1.633
14	1.738	1.923	1.742	1.733	1.629	1.362
15	2.156	1.921	2.000	2.218	2.060	1.613
17	1.631	1.838	1.431	1.660	1.644	1.342
18	1.723	1.844	1.539	1.753	1.688	1.491
19	1.728	1.870	1.477	1.824	1.699	1.602
20	1.605	1.924	1.417	1.688	1.740	1.415
21	1.626	1.809	1.490	1.630	1.534	1.519
22	1.664	1.851	1.497	1.704	1.651	1.398
23	1.751	1.855	1.405	1.840	1.741	1.519
24	1.597	1.490	1.365	1.664	1.560	1.415

REVIC : Regression Output:

Constant	0.435903		
Std Err of Y Est	0.164697	SSE =	0.515375
R Squared	0.42555	SST =	0.897162
No. of Observations	21	SSR =	0.381787
Degrees of Freedom	19		
		F =	14.07511
X Coefficient(s)	0.643387		
Std Err of Coef.	0.171493	F(.95,1,19)	4.38
		F(.99,1,19)	8.19

SPQR/20 : Regression Output:

Constant	0.359679		
Std Err of Y Est	0.17594	SSE =	0.588145
R Squared	0.344438	SST =	0.897162
No. of Observations	21	SSR =	0.309017
Degrees of Freedom	19		
		F =	9.982782
X Coefficient(s)	0.644941		
Std Err of Coef.	0.204124	F(.95,1,19)	4.38
		F(.99,1,19)	8.19

System 4: Regression Output:

Constant	0.478035		
Std Err of Y Est	0.155078	SSE =	0.456933
R Squared	0.490691	SST =	0.897162
No. of Observations	21	SSR =	0.44023
Degrees of Freedom	19		
		F =	18.30547
X Coefficient(s)	0.680146		
Std Err of Coef.	0.158969	F(.95,1,19)	4.38
		F(.99,1,19)	8.19

SEER : Regression Output:

Constant	0.461073		
Std Err of Y Est	0.165378	SSE =	0.519645
R Squared	0.420791	SST =	0.897162
No. of Observations	21	SSR =	0.377517
Degrees of Freedom	19		
		F =	13.80334
X Coefficient(s)	0.634176		
Std Err of Coef.	0.170694	F(.95,1,19)	4.38
		F(.99,1,19)	8.19

Price-S : Regression Output:

Constant	0.314083		
Std Err of Y Est	0.174601	SSE =	0.579226
R Squared	0.35438	SST =	0.897162
No. of Observations	21	SSR =	0.317937
Degrees of Freedom	19		
		F =	10.42909
X Coefficient(s)	0.722572		
Std Err of Coef.	0.223747	F(.95,1,19)	4.38
		F(.99,1,19)	8.19

Appendix C: Wilcoxon Test

Wilcoxon Test - REVIC

Project #	REVIC	Actuals	Delta	Rank +	Rank -
1	32.3	29.0	3.3	1	
2	19.7	27.0	-7.3		3
3	28.0	18.0	10	7	
4	25.6	33.0	-7.4		4
5	25.9	13.0	12.9	8	
8	61.1	68.0	-6.9		2
9	62.0	71.0	-9		5
10	141.9	84.0	57.9	19	
11	51.9	22.0	29.9	18	
12	61.8	75.0	-13.2		9
13	71.1	43.0	28.1	17	
14	54.7	23.0	31.7	20	
15	143.3	41.0	102.3	21	
17	42.8	22.0	20.8	13	
18	52.8	31.0	21.8	15	
19	53.5	40.0	13.5	11	
20	40.3	26.0	14.3	12	
21	42.3	33.0	9.3	6	
22	46.1	25.0	21.1	14	
23	56.3	33.0	23.3	16	
24	39.5	26.0	13.5	10	
			SUMS	208	23

Wilcoxon T = 23

H_o = center of paired differences of population distributions = 0
(ie no differences in the distributions)

$\mu(T) = n(n+1)/4 =$ 115.5 Number of non-zero deltas >=20
therefore approximates normal curve

$\sigma^2(T) = n(n+1)(2n+1)/24 =$ 827.75

Test stat = $(T - \mu)/\sigma =$ -3.21508

z (.95) = 1.645
z (.99) = 2.33

Results of test = fail at both levels

Wilcoxon Test - SPQR/20

Project #	SPQR/20	Actuals	Delta	Rank +	Rank -
1	36.9	29.0	7.91	4	
2	35.7	27.0	8.71	5	
3	29.7	18.0	11.67	6	
4	28.2	33.0	-4.77		2
5	39.9	13.0	26.9	8	
8	86.1	68.0	18.13	7	
9	109.9	71.0	38.93	14	
10	116.1	84.0	32.07	10	
11	73.0	22.0	50.99	18	
12	77.1	75.0	2.08	1	
13	103.1	43.0	60.13	20	
14	83.7	23.0	60.69	21	
15	83.4	41.0	42.4	15	
17	68.9	22.0	46.9	17	
18	69.9	31.0	38.88	13	
19	74.2	40.0	34.2	11	
20	84.0	26.0	57.98	19	
21	64.5	33.0	31.49	9	
22	70.9	25.0	45.88	16	
23	71.6	33.0	38.59	12	
24	30.9	26.0	4.93	3	
			SUMS	229	2

Wilcoxon T = 2

H_o = center of paired differences of population distributions = 0
(ie no differences in the distributions)

$\mu(T) = n(n+1)/4 =$ 115.5 Number of non-zero deltas >=20
therefore approximates normal curve

$\sigma^2(T) = n(n+1)(2n+1)/24 =$ 827.75

Test stat = $(T - \mu)/\sigma =$ -3.94499

z (.95) = 1.645
z (.99) = 2.33

Results of test = fail at both levels

Wilcoxon Test - System-4

Project #	System 4	Actuals	Delta	Rank +	Rank -
1	28.5	29.0	-0.5		2.5
2	19.0	27.0	-8		11
3	23.0	18.0	5	7	
4	24.1	33.0	-8.9		13
5	13.5	13.0	0.5	2.5	
8	44.6	68.0	-23.4		17
9	44.9	71.0	-26.1		18
10	107.3	84.0	23.3	16	
11	30.4	22.0	8.4	12	
12	45.9	75.0	-29.1		19
13	52.6	43.0	9.6	14	
14	55.2	23.0	32.2	20	
15	99.9	41.0	58.9	21	
17	27.0	22.0	5	8	
18	34.6	31.0	3.6	6	
19	30.0	40.0	-10		15
20	26.1	26.0	0.1	1	
21	30.9	33.0	-2.1		4
22	31.4	25.0	6.4	9	
23	25.4	33.0	-7.6		10
24	23.2	26.0	-2.8		5
			SUMS	116.5	114.5

Wilcoxon T = 114.5

H_o = center of paired differences of population distributions = 0
(ie no differences in the distributions)

$\mu(T) = n(n+1)/4 =$ 115.5 Number of non-zero deltas >=20
therefore approximates normal curve

$\sigma^2(T) = n(n+1)(2n+1)/24 =$ 827.75

Test stat = $(T - \mu)/\sigma =$ -0.03476

z (.95) = 1.645
z (.99) = 2.33

Results of test = pass at both levels

Wilcoxon Test - SEER

Project #	SEER	Actuals	Delta	Rank +	Rank -
1	31.6	29.0	2.56		1
2	22.2	27.0	-4.77		4
3	30.8	18.0	12.79	6	
4	30.3	33.0	-2.7		3
5	26.8	13.0	13.82	7	
8	66.7	68.0	-1.35		5
9	62.7	71.0	-8.35	11	
10	158.0	84.0	73.97	20	
11	55.3	22.0	33.27	18	
12	68.0	75.0	-7.01		9
13	62.5	43.0	19.46	8	
14	54.1	23.0	31.12	17	
15	165.2	41.0	124.16	21	
17	45.7	22.0	23.7	13	
18	56.7	31.0	25.66	15	
19	66.7	40.0	26.69	14	
20	48.7	26.0	22.73	12	
21	42.6	33.0	9.64	2	
22	50.6	25.0	25.56	16	
23	69.2	33.0	36.22	19	
24	46.1	26.0	20.1	10	
			SUMS	209	22

Wilcoxon T = 22

H_o = center of paired differences of population distributions = 0
(ie no differences in the distributions)

$\mu(T) = n(n+1)/4 =$ 115.5 Number of non-zero deltas >=20
therefore approximates normal curve

$\sigma^2(T) = n(n+1)(2n+1)/24 =$ 827.75

Test stat = $(T - \mu)/\sigma =$ -3.24984

z (.95) = 1.645
z (.99) = 2.33

Results of test = fail at both levels

Wilcoxon Test - Price-S

Project #	PRICE-S	Actuals	Delta	Rank +	Rank -
1	45.7	29.0	16.7	8	
2	22.8	27.0	-4.2		3
3	30.4	18.0	12.4	6	
4	36.9	33.0	3.9	2	
5	29.3	13.0	16.3	7	
8	48.0	68.0	-20		14
9	43.9	71.0	-27.1		18
10	131.6	84.0	47.6	20	
11	39.2	22.0	17.2	9	
12	53.2	75.0	-21.8		15
13	62.7	43.0	19.7	12	
14	42.6	23.0	19.6	11	
15	114.9	41.0	73.9	21	
17	44.1	22.0	22.1	16.5	
18	48.8	31.0	17.8	10	
19	50.0	40.0	10	4	
20	54.9	26.0	28.9	19	
21	34.2	33.0	1.2	1	
22	44.8	25.0	19.8	13	
23	55.1	33.0	22.1	16.5	
24	36.3	26.0	10.3	5	
			SUMS	181	50

Wilcoxon T = 50

H_o = center of paired differences of population distributions = 0
(ie no differences in the distributions)

$\mu(T) = n(n+1)/4 =$ 115.5 Number of non-zero deltas >=20 therefore approximates normal curve

$\sigma^2(T) = n(n+1)(2n+1)/24 =$ 827.75

Test stat = $(T - \mu)/\sigma =$ -2.27663

z (.95) = 1.645
z (.99) = 2.33

Results of test = fail at .95, pass at .99

Appendix D: Friedman Rank Sum

(Ranks in Parentheses)

Proj #	Actuals	REVIC	SPQR/20	System 4	SEER	PRICE-S
1	29 (2)	32.3 (4)	36.9 (5)	28.5 (1)	31.6 (3)	45.7 (6)
2	27 (5)	19.7 (2)	35.7 (6)	19.0 (1)	22.2 (3)	22.8 (4)
3	18 (1)	28.0 (3)	29.7 (4)	23.0 (2)	30.8 (6)	30.4 (5)
4	33 (5)	25.6 (2)	28.2 (3)	24.1 (1)	30.3 (4)	36.9 (6)
5	13 (1)	25.9 (3)	39.9 (6)	13.5 (2)	26.8 (4)	29.3 (5)
8	68 (5)	61.1 (3)	86.1 (6)	44.6 (1)	66.7 (4)	48.0 (2)
9	71 (5)	62.0 (3)	109.9 (4)	44.9 (2)	62.7 (4)	43.9 (1)
10	84 (1)	141.9 (5)	116.1 (3)	107.3 (2)	158.0 (6)	131.6 (4)
11	22 (1)	51.9 (4)	73.0 (6)	30.4 (2)	55.3 (5)	39.2 (3)
12	75 (5)	61.8 (3)	77.1 (6)	45.9 (1)	68.0 (4)	53.2 (2)
13	43 (1)	71.1 (5)	103.1 (6)	52.6 (2)	62.5 (3)	62.7 (4)
14	23 (1)	54.7 (4)	83.7 (6)	55.2 (5)	54.1 (3)	42.6 (2)
15	41 (1)	143.3 (5)	83.4 (2)	99.9 (3)	165.2 (6)	114.9 (4)
17	22 (1)	42.8 (3)	68.9 (6)	27.0 (2)	45.7 (5)	44.1 (4)
18	31 (1)	52.8 (4)	69.9 (6)	34.6 (2)	56.7 (5)	43.8 (3)
19	40 (2)	53.5 (4)	74.2 (6)	30.0 (1)	66.7 (5)	50.0 (3)
20	26 (1)	40.3 (3)	84.0 (6)	26.1 (2)	48.7 (4)	54.9 (5)
21	33 (2)	42.3 (4)	64.5 (6)	30.9 (1)	42.6 (5)	34.2 (3)
22	25 (1)	46.1 (4)	70.9 (6)	31.4 (2)	50.6 (5)	44.8 (3)
23	33 (2)	56.3 (4)	71.6 (6)	25.4 (1)	69.2 (5)	55.1 (3)
24	26 (2)	39.5 (5)	30.9 (3)	23 (1)	46.1 (6)	36.3 (4)
Rank Sums	46	77	108	37	95	76

Avg for 5 72.6

H_o: All data sets are the same
H_a: Not all data sets are the same

S = 47.14966

χ^2 (5,.05) = 11
χ^2 (5,.01) = 15

Fails, therefore not equivalent at both levels

	Actuals	REVIC	SPQR/20	System 4	SEER	PRICE-S
Rank Sums	46	77	108	37	95	76
R1 vs Rn =		31	62	9	49	30

H_o: Both compared data sets are equal
H_a: The compared data sets are not equal

m (.05,5,.5) = 2.51, therefore test statistic m* = 30.4321
m (.01,5,.5) = 3.06, therefore test statistic m* = 37.1005

at .05 level:	Different	Different	Same	Different	Same
at .01 level:	Same	Different	Same	Different	Same

Appendix E: Percentage Method

Project #	REVIC	SPQR/20	System 4	SEER	PRICE-S
1	1.11	1.27	0.98	1.09	1.58
2	0.73	1.32	0.70	0.82	0.84
3	1.56	1.65	1.28	1.71	1.69
4	0.78	0.86	0.73	0.92	1.12
5	1.99	3.07	1.04	2.06	2.25
8	0.90	1.27	0.66	0.98	0.71
9	0.87	1.55	0.63	0.88	0.62
10	1.69	1.38	1.28	1.88	1.57
11	2.36	3.32	1.38	2.51	1.78
12	0.82	1.03	0.61	0.91	0.71
13	1.65	2.40	1.22	1.45	1.46
14	2.38	3.64	2.40	2.35	1.85
15	3.50	2.03	2.44	4.03	2.80
17	1.95	3.13	1.23	2.08	2.00
18	1.70	2.25	1.12	1.83	1.57
19	1.34	1.86	0.75	1.67	1.25
20	1.55	3.23	1.00	1.87	2.11
21	1.28	1.95	0.94	1.29	1.04
22	1.84	2.84	1.26	2.02	1.79
23	1.71	2.17	0.77	2.10	1.67
24	1.52	1.19	0.89	1.77	1.40

Note: The values above are the estimated schedule divided by the actual schedule. For example, on project #1, REVIC estimated a schedule 11% longer than the actual schedule.

Is the estimate within 30%?

Project #	REVIC	SPQR/20	System 4	SEER	PRICE-S
1	Pass	Pass	Pass	Pass	Fail
2	Pass	Fail	Pass	Pass	Pass
3	Fail	Fail	Pass	Fail	Fail
4	Pass	Pass	Pass	Pass	Pass
5	Fail	Fail	Pass	Fail	Fail
8	Pass	Pass	Fail	Pass	Pass
9	Pass	Fail	Fail	Pass	Fail
10	Fail	Fail	Pass	Fail	Fail
11	Fail	Fail	Fail	Fail	Fail
12	Pass	Pass	Fail	Pass	Pass
13	Fail	Fail	Pass	Fail	Fail
14	Fail	Fail	Fail	Fail	Fail
15	Fail	Fail	Fail	Fail	Fail
17	Fail	Fail	Pass	Fail	Fail
18	Fail	Fail	Pass	Fail	Fail
19	Fail	Fail	Pass	Fail	Pass
20	Fail	Fail	Pass	Fail	Fail
21	Pass	Fail	Pass	Pass	Pass
22	Fail	Fail	Pass	Fail	Fail
23	Fail	Fail	Pass	Fail	Fail
24	Fail	Pass	Pass	Fail	Fail

	REVIC	SPQR/20	System 4	SEER	PRICE-S
	Pass	Pass	Pass	Pass	Pass
	REVIC	SPQR/20	System 4	SEER	PRICE-S
	Fail	Fail	Fail	Fail	Fail
Pass	7	5	15	7	6
Fail	14	16	6	14	15
% Pass	33.3%	23.8%	71.4%	33.3%	28.6%

Is the estimate within 20%?

Project #	REVIC	SPQR/20	System 4	SEER	PRICE-S
1	Pass	Fail	Pass	Pass	Fail
2	Fail	Fail	Fail	Pass	Pass
3	Fail	Fail	Fail	Fail	Fail
4	Fail	Pass	Fail	Pass	Pass
5	Fail	Fail	Pass	Fail	Fail
8	Pass	Fail	Fail	Pass	Fail
9	Pass	Fail	Fail	Pass	Fail
10	Fail	Fail	Fail	Fail	Fail
11	Fail	Fail	Fail	Fail	Fail
12	Pass	Pass	Fail	Pass	Fail
13	Fail	Fail	Fail	Fail	Fail
14	Fail	Fail	Fail	Fail	Fail
15	Fail	Fail	Fail	Fail	Fail
17	Fail	Fail	Fail	Fail	Fail
18	Fail	Fail	Pass	Fail	Fail
19	Fail	Fail	Fail	Fail	Fail
20	Fail	Fail	Pass	Fail	Fail
21	Fail	Fail	Pass	Fail	Pass
22	Fail	Fail	Fail	Fail	Fail
23	Fail	Fail	Fail	Fail	Fail
24	Fail	Pass	Pass	Fail	Fail

	REVIC	SPQR/20	System 4	SEER	PRICE-S
	Pass	Pass	Pass	Pass	Pass
	REVIC	SPQR/20	System 4	SEER	PRICE-S
	Fail	Fail	Fail	Fail	Fail
Pass	4	3	6	6	3
Fail	17	18	15	15	18
% Pass	19.0%	14.3%	28.6%	28.6%	14.3%

Appendix F: Adjusted Data

Raw Data

Project #	REVIC	SPQR/20	System 4	SEER	PRICE-S
1	1.11	1.27	0.98	1.09	1.58
2	0.73	1.32	0.70	0.82	0.84
3	1.56	1.65	1.28	1.71	1.69
4	0.78	0.86	0.73	0.92	1.12
5	1.99	3.07	1.04	2.06	2.25
8	0.90	1.27	0.66	0.98	0.71
9	0.87	1.55	0.63	0.88	0.62
10	1.69	1.38	1.28	1.88	1.57
11	2.36	3.32	1.38	2.51	1.78
12	0.82	1.03	0.61	0.91	0.71
13	1.65	2.40	1.22	1.45	1.46
14	2.38	3.64	2.40	2.35	1.85
15	3.50	2.03	2.44	4.03	2.80
17	1.95	3.13	1.23	2.08	2.00
18	1.70	2.25	1.12	1.83	1.57
19	1.34	1.86	0.75	1.67	1.25
20	1.55	3.23	1.00	1.87	2.11
21	1.28	1.95	0.94	1.29	1.04
22	1.84	2.84	1.26	2.02	1.79
23	1.71	2.17	0.77	2.10	1.67
24	1.52	1.19	0.89	1.77	1.40
Avg	1.58	2.07	1.11	1.73	1.51

Adjustment	63.2%	48.4%	90.1%	58.0%	66.0%
Project #	REVIC	SPQR/20	System 4	SEER	PRICE-S
1	0.70	0.62	0.89	0.63	1.04
2	0.46	0.64	0.63	0.48	0.56
3	0.98	0.80	1.15	0.99	1.11
4	0.49	0.41	0.66	0.53	0.74
5	1.26	1.49	0.94	1.20	1.49
8	0.57	0.61	0.59	0.57	0.47
9	0.55	0.75	0.57	0.51	0.41
10	1.07	0.67	1.15	1.09	1.03
11	1.49	1.61	1.25	1.46	1.18
12	0.52	0.50	0.55	0.53	0.47
13	1.05	1.16	1.10	0.84	0.96
14	1.50	1.76	2.16	1.36	1.22
15	2.21	0.98	2.20	2.33	1.85
17	1.23	1.52	1.11	1.20	1.32
18	1.08	1.09	1.01	1.06	1.04
19	0.85	0.90	0.68	0.97	0.83
20	0.98	1.56	0.90	1.09	1.39
21	0.81	0.95	0.84	0.75	0.68
22	1.17	1.37	1.13	1.17	1.18
23	1.08	1.05	0.69	1.22	1.10
24	0.96	0.58	0.80	1.03	0.92

Is the estimate within 30%?

Project #	REV	SPQR	Syst	SER	PRICE
1	Pass	Fail	Pass	Fail	Pass
2	Fail	Fail	Fail	Fail	Fail
3	Pass	Pass	Pass	Pass	Pass
4	Fail	Fail	Fail	Fail	Pass
5	Pass	Fail	Pass	Pass	Fail
8	Fail	Fail	Fail	Fail	Fail
9	Fail	Pass	Fail	Fail	Fail
10	Pass	Fail	Pass	Pass	Pass
11	Fail	Fail	Pass	Fail	Pass
12	Fail	Fail	Fail	Fail	Fail
13	Pass	Pass	Pass	Pass	Pass
14	Fail	Fail	Fail	Fail	Pass
15	Fail	Pass	Fail	Fail	Fail
17	Pass	Fail	Pass	Pass	Fail
18	Pass	Pass	Pass	Pass	Pass
19	Pass	Pass	Fail	Pass	Pass
20	Pass	Fail	Pass	Pass	Fail
21	Pass	Pass	Pass	Pass	Fail
22	Pass	Fail	Pass	Pass	Pass
23	Pass	Pass	Fail	Pass	Pass
24	Pass	Fail	Pass	Pass	Pass
Pass	13	8	12	12	12
Fail	8	13	9	9	9
% Pass	61.9%	38.1%	57.1%	57.1%	57.1%

Adjustment	90.0%	90.0%	100.0%	90.0%	90.0%
Project #	REVIC	SPQR/20	System 4	SEER	PRICE-S
1	1.00	1.15	0.98	0.98	1.42
2	0.66	1.19	0.70	0.74	0.76
3	1.40	1.48	1.28	1.54	1.52
4	0.70	0.77	0.73	0.83	1.01
5	1.79	2.76	1.04	1.86	2.03
8	0.81	1.14	0.66	0.88	0.64
9	0.79	1.39	0.63	0.79	0.56
10	1.52	1.24	1.28	1.69	1.41
11	2.12	2.99	1.38	2.26	1.60
12	0.74	0.92	0.61	0.82	0.64
13	1.49	2.16	1.22	1.31	1.31
14	2.14	3.27	2.40	2.12	1.67
15	3.15	1.83	2.44	3.63	2.52
17	1.75	2.82	1.23	1.87	1.80
18	1.53	2.03	1.12	1.64	1.42
19	1.20	1.67	0.75	1.50	1.13
20	1.40	2.91	1.00	1.69	1.90
21	1.15	1.76	0.94	1.16	0.93
22	1.66	2.55	1.26	1.82	1.61
23	1.54	1.95	0.77	1.89	1.50
24	1.37	1.07	0.89	1.60	1.26

Is the estimate within 30%?

Project #	REV	SPQR	Syst	SER	PRICE
1	Pass	Pass	Pass	Pass	Fail
2	Fail	Pass	Pass	Pass	Pass
3	Fail	Fail	Pass	Fail	Fail
4	Fail	Pass	Pass	Pass	Pass
5	Fail	Fail	Pass	Fail	Fail
8	Pass	Pass	Fail	Pass	Fail
9	Pass	Fail	Fail	Pass	Fail
10	Fail	Pass	Pass	Fail	Fail
11	Fail	Fail	Fail	Fail	Fail
12	Pass	Pass	Fail	Pass	Fail
13	Fail	Fail	Pass	Fail	Fail
14	Fail	Fail	Fail	Fail	Fail
15	Fail	Fail	Fail	Fail	Fail
17	Fail	Fail	Pass	Fail	Fail
18	Fail	Fail	Pass	Fail	Fail
19	Pass	Fail	Pass	Fail	Pass
20	Fail	Fail	Pass	Fail	Fail
21	Pass	Fail	Pass	Pass	Pass
22	Fail	Fail	Pass	Fail	Fail
23	Fail	Fail	Pass	Fail	Fail
24	Fail	Pass	Pass	Fail	Pass
Pass	6	7	15	7	5
Fail	15	14	6	14	16
% Pass	28.6%	33.3%	71.4%	33.3%	23.8%

Adjustment	80.0%	80.0%	100.0%	80.0%	80.0%
Project #	REVIC	SPQR/20	System 4	SEER	PRICE-S
1	0.89	1.02	0.98	0.87	1.26
2	0.58	1.06	0.70	0.66	0.68
3	1.24	1.32	1.28	1.37	1.35
4	0.62	0.68	0.73	0.73	0.89
5	1.59	2.46	1.04	1.65	1.80
8	0.72	1.01	0.66	0.78	0.56
9	0.70	1.24	0.63	0.71	0.49
10	1.35	1.11	1.28	1.50	1.25
11	1.89	2.65	1.38	2.01	1.43
12	0.66	0.82	0.61	0.73	0.57
13	1.32	1.92	1.22	1.16	1.17
14	1.90	2.91	2.40	1.88	1.48
15	2.80	1.63	2.44	3.22	2.24
17	1.56	2.51	1.23	1.66	1.60
18	1.36	1.80	1.12	1.46	1.26
19	1.07	1.48	0.75	1.33	1.00
20	1.24	2.58	1.00	1.50	1.69
21	1.03	1.56	0.94	1.03	0.83
22	1.48	2.27	1.26	1.62	1.43
23	1.36	1.74	0.77	1.68	1.34
24	1.22	0.95	0.89	1.42	1.12

Is the estimate within 30%?

Project #	REV	SPQR	Syst	SER	PRICE
1	Pass	Pass	Pass	Pass	Pass
2	Fail	Pass	Pass	Fail	Fail
3	Pass	Fail	Pass	Fail	Fail
4	Fail	Fail	Pass	Pass	Pass
5	Fail	Fail	Pass	Fail	Fail
8	Pass	Pass	Fail	Pass	Fail
9	Fail	Pass	Fail	Pass	Fail
10	Fail	Pass	Pass	Fail	Pass
11	Fail	Fail	Fail	Fail	Fail
12	Fail	Pass	Fail	Pass	Fail
13	Fail	Fail	Pass	Pass	Pass
14	Fail	Fail	Fail	Fail	Fail
15	Fail	Fail	Fail	Fail	Fail
17	Fail	Fail	Pass	Fail	Fail
18	Fail	Fail	Pass	Fail	Pass
19	Pass	Fail	Pass	Fail	Pass
20	Pass	Fail	Pass	Fail	Fail
21	Pass	Fail	Pass	Pass	Pass
22	Fail	Fail	Pass	Fail	Fail
23	Fail	Fail	Pass	Fail	Fail
24	Pass	Pass	Pass	Fail	Pass
Pass	7	7	15	7	8
Fail	14	14	6	14	13
% Pass	33.3%	33.3%	71.4%	33.3%	38.1%

Adjustment	70.0%	70.0%	100.0%	70.0%	70.0%
Project #	REVIC	SPQR/20	System 4	SEER	PRICE-S
1	0.78	0.89	0.98	0.76	1.10
2	0.51	0.93	0.70	0.58	0.59
3	1.09	1.15	1.28	1.20	1.18
4	0.54	0.60	0.73	0.64	0.78
5	1.39	2.15	1.04	1.44	1.38
8	0.63	0.89	0.66	0.69	0.49
9	0.61	1.08	0.63	0.62	0.43
10	1.18	0.97	1.28	1.32	1.10
11	1.65	2.32	1.38	1.76	1.25
12	0.58	0.72	0.61	0.63	0.50
13	1.16	1.68	1.22	1.02	1.02
14	1.66	2.55	2.40	1.65	1.30
15	2.45	1.42	2.44	2.82	1.96
17	1.36	2.19	1.23	1.45	1.40
18	1.19	1.58	1.12	1.28	1.10
19	0.94	1.30	0.75	1.17	0.88
20	1.08	2.26	1.00	1.31	1.48
21	0.90	1.37	0.94	0.90	0.73
22	1.29	1.98	1.26	1.42	1.25
23	1.19	1.52	0.77	1.47	1.17
24	1.06	0.83	0.89	1.24	0.98

Is the estimate within 30%?

Project #	REV	SPQR	Syst	SER	PRICE
1	Pass	Pass	Pass	Pass	Pass
2	Fail	Pass	Pass	Fail	Fail
3	Pass	Pass	Pass	Pass	Pass
4	Fail	Fail	Pass	Fail	Pass
5	Fail	Fail	Pass	Fail	Fail
8	Fail	Pass	Fail	Fail	Fail
9	Fail	Pass	Fail	Fail	Fail
10	Pass	Pass	Pass	Fail	Pass
11	Fail	Fail	Fail	Fail	Pass
12	Fail	Pass	Fail	Fail	Fail
13	Pass	Fail	Pass	Pass	Pass
14	Fail	Fail	Fail	Fail	Pass
15	Fail	Fail	Fail	Fail	Fail
17	Fail	Fail	Pass	Fail	Fail
18	Pass	Fail	Pass	Pass	Pass
19	Pass	Pass	Pass	Pass	Pass
20	Pass	Fail	Pass	Fail	Fail
21	Pass	Fail	Pass	Pass	Pass
22	Pass	Fail	Pass	Fail	Pass
23	Pass	Fail	Pass	Fail	Pass
24	Pass	Pass	Pass	Pass	Pass
Pass	11	9	15	7	13
Fail	10	12	6	14	8
% Pass	52.4%	42.9%	71.4%	33.3%	61.9%

Adjustment	60.0%	60.0%	100.0%	60.0%	60.0%
Project #	REVIC	SPQR/20	System 4	SEER	PRICE-S
1	0.67	0.76	0.98	0.65	0.95
2	0.44	0.79	0.70	0.49	0.51
3	0.93	0.99	1.28	1.03	1.01
4	0.47	0.51	0.73	0.55	0.67
5	1.20	1.84	1.04	1.24	1.35
8	0.54	0.76	0.66	0.59	0.42
9	0.52	0.93	0.63	0.53	0.37
10	1.01	0.83	1.28	1.13	0.94
11	1.42	1.99	1.38	1.51	1.07
12	0.49	0.62	0.61	0.54	0.43
13	0.99	1.44	1.22	0.87	0.87
14	1.43	2.18	2.40	1.41	1.11
15	2.10	1.22	2.44	2.42	1.68
17	1.17	1.88	1.23	1.25	1.20
18	1.02	1.35	1.12	1.10	0.94
19	0.80	1.11	0.75	1.00	0.75
20	0.93	1.94	1.00	1.12	1.27
21	0.77	1.17	0.94	0.78	0.62
22	1.11	1.70	1.26	1.21	1.08
23	1.02	1.30	0.77	1.26	1.00
24	0.91	0.71	0.89	1.06	0.84

Is the estimate within 30%?

Project #	REV	SPQR	Syst	SER	PRICE
1	Fail	Pass	Pass	Fail	Pass
2	Fail	Pass	Pass	Fail	Fail
3	Pass	Pass	Pass	Pass	Pass
4	Fail	Fail	Pass	Fail	Fail
5	Pass	Fail	Pass	Pass	Fail
8	Fail	Pass	Fail	Fail	Fail
9	Fail	Pass	Fail	Fail	Fail
10	Pass	Pass	Pass	Pass	Pass
11	Fail	Fail	Fail	Fail	Pass
12	Fail	Fail	Fail	Fail	Fail
13	Pass	Fail	Pass	Pass	Pass
14	Fail	Fail	Fail	Fail	Pass
15	Fail	Pass	Fail	Fail	Fail
17	Pass	Fail	Pass	Pass	Pass
18	Pass	Fail	Pass	Pass	Pass
19	Pass	Pass	Pass	Pass	Pass
20	Pass	Fail	Pass	Pass	Pass
21	Pass	Pass	Pass	Pass	Fail
22	Pass	Fail	Pass	Pass	Pass
23	Pass	Fail	Pass	Pass	Pass
24	Pass	Pass	Pass	Pass	Pass
Pass	12	10	15	12	13
Fail	9	11	6	9	8
% Pass	57.1%	47.6%	71.4%	57.1%	61.9%

Adjustment	50.0%	50.0%	100.0%	50.0%	50.0%
Project #	REVIC	SPQR/20	System 4	SEER	PRICE-S
1	0.56	0.64	0.98	0.54	0.79
2	0.36	0.66	0.70	0.41	0.42
3	0.78	0.82	1.28	0.86	0.84
4	0.39	0.43	0.73	0.46	0.56
5	1.00	1.53	1.04	1.03	1.13
8	0.45	0.63	0.66	0.49	0.35
9	0.44	0.77	0.63	0.44	0.31
10	0.84	0.69	1.28	0.94	0.78
11	1.18	1.66	1.38	1.26	0.89
12	0.41	0.51	0.61	0.45	0.35
13	0.83	1.20	1.22	0.73	0.73
14	1.19	1.82	2.40	1.18	0.93
15	1.75	1.02	2.44	2.01	1.40
17	0.97	1.57	1.23	1.04	1.00
18	0.85	1.13	1.12	0.91	0.79
19	0.67	0.93	0.75	0.83	0.63
20	0.78	1.62	1.00	0.94	1.06
21	0.64	0.98	0.94	0.65	0.52
22	0.92	1.42	1.26	1.01	0.90
23	0.85	1.08	0.77	1.05	0.83
24	0.76	0.59	0.89	0.89	0.70

Is the estimate within 30%?

Project #	REV	SPQR	Syst	SER	PRICE
1	Fail	Fail	Pass	Fail	Pass
2	Fail	Fail	Pass	Fail	Fail
3	Pass	Pass	Pass	Pass	Pass
4	Fail	Fail	Pass	Fail	Fail
5	Pass	Fail	Pass	Pass	Pass
8	Fail	Fail	Fail	Fail	Fail
9	Fail	Pass	Fail	Fail	Fail
10	Pass	Fail	Pass	Pass	Pass
11	Pass	Fail	Fail	Pass	Pass
12	Fail	Fail	Fail	Fail	Fail
13	Pass	Pass	Pass	Pass	Pass
14	Pass	Fail	Fail	Pass	Pass
15	Fail	Pass	Fail	Fail	Fail
17	Pass	Fail	Pass	Pass	Pass
18	Pass	Pass	Pass	Pass	Pass
19	Fail	Pass	Pass	Pass	Fail
20	Pass	Fail	Pass	Pass	Pass
21	Fail	Pass	Pass	Fail	Fail
22	Pass	Fail	Pass	Pass	Pass
23	Pass	Pass	Pass	Pass	Pass
24	Pass	Fail	Pass	Pass	Fail
Pass	12	8	15	13	12
Fail	9	13	6	8	9
% Pass	57.1%	38.1%	71.4%	61.9%	57.1%

Adjustment	40.0%	40.0%	100.0%	40.0%	40.0%
Project #	REVIC	SPQR/20	System 4	SEER	PRICE-S
1	0.45	0.51	0.98	0.44	0.63
2	0.29	0.53	0.70	0.33	0.34
3	0.62	0.66	1.28	0.68	0.68
4	0.31	0.34	0.73	0.37	0.45
5	0.80	1.23	1.04	0.83	0.90
8	0.36	0.51	0.66	0.39	0.28
9	0.35	0.62	0.63	0.35	0.25
10	0.68	0.55	1.28	0.75	0.63
11	0.94	1.33	1.38	1.00	0.71
12	0.33	0.41	0.61	0.36	0.28
13	0.66	0.96	1.22	0.58	0.58
14	0.95	1.46	2.40	0.94	0.74
15	1.40	0.81	2.44	1.61	1.12
17	0.78	1.25	1.23	0.83	0.80
18	0.68	0.90	1.12	0.73	0.63
19	0.54	0.74	0.75	0.67	0.50
20	0.62	1.29	1.00	0.75	0.84
21	0.51	0.78	0.94	0.52	0.41
22	0.74	1.13	1.26	0.81	0.72
23	0.68	0.87	0.77	0.84	0.67
24	0.61	0.48	0.89	0.71	0.56

Is the estimate within 30%?

Project #	REV	SPQR	Syst	SER	PRICE
1	Fail	Fail	Pass	Fail	Fail
2	Fail	Fail	Pass	Fail	Fail
3	Fail	Fail	Pass	Fail	Fail
4	Fail	Fail	Pass	Fail	Fail
5	Pass	Pass	Pass	Pass	Pass
8	Fail	Fail	Fail	Fail	Fail
9	Fail	Fail	Fail	Fail	Fail
10	Fail	Fail	Pass	Pass	Fail
11	Pass	Fail	Fail	Pass	Pass
12	Fail	Fail	Fail	Fail	Fail
13	Fail	Pass	Pass	Fail	Fail
14	Pass	Fail	Fail	Pass	Pass
15	Fail	Pass	Fail	Fail	Pass
17	Pass	Pass	Pass	Pass	Pass
18	Fail	Pass	Pass	Pass	Fail
19	Fail	Pass	Pass	Fail	Fail
20	Fail	Pass	Pass	Pass	Pass
21	Fail	Pass	Pass	Fail	Fail
22	Pass	Pass	Pass	Pass	Pass
23	Fail	Pass	Pass	Pass	Fail
24	Fail	Fail	Pass	Pass	Fail
Pass	5	10	15	10	7
Fail	16	11	6	11	14
% Pass	23.8%	47.6%	71.4%	47.6%	33.3%

Bibliography

1. Air Force Studies Board. Adapting Software Development Policies to Modern Technology. Washington: National Academy Press, 1989.

2. Blalock, Crystal D. An Analysis of Schedule Determination in Software Program Development and Software Development Estimation Models. MS Thesis, AFIT/GCA/LSY/88S-2. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright Patterson AFB OH, September 1988 (DTIC #AD-A111 204).

3. Boehm, Barry W. "A Spiral Model of Software Development and Enhancement," Tutorial: Software Engineering Project Management. Washington DC: IEEE, 1988

4. -----. Software Engineering Economics. Englewood Cliffs NJ: Prentice-Hall Inc., 1981.

5. -----. "Understanding and Controlling Software Estimates," Journal of Parametrics, VII: 32-67 (March 1988).

6. ----- and Philip N. Papaccio. "Understanding and Controlling Software Costs," IEEE Transactions on Software Engineering, 14: 1462-1477 (Oct 1988).

7. Computer Economics, Inc. Announcing System-4. Marina del Ray CA: CEI, Inc., 1989.

8. Conte, Samuel D., H. E. Dunsmore, and V. Y. Shen. Software Engineering Metrics and Models. Menlo Park: Benjamin/Cummings Publishing Co., 1986.

9. Daft, Richard L. and Richard M. Steers. Organizations - A Micro/Macro Approach. Glenview IL: Scott, Foresman and Co., 1986.

10. Davis, Alan M., Edward H. Bersoff, and Edward R. Comer. "A Strategy for Comparing Alternative Software Development Life Cycle Models," IEEE Transactions on Software Engineering, 14: 1453-1461 (October 1988).

11. Department of Defense. Defense System Software Development. Military Standard 2167A. Washington: DoD, 29 February 1988.

12. Duncan, William. "Get Out from Under," Computerworld: 68-73 (February 1987).

13. Ferens, Daniel V. Defense System Software Project Management. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright Patterson AFB OH, 1990.

14. ----. "Software Schedule Estimation: The Third Wave," Journal of Parametrics, X: 41-52 (February 1990).

15. Firesmith, Donald G. "Should the DoD Mandate a Standard Software Development Process?" DS&E: 56-59 (July 1987).

16. Funch, Paul, MIT Research Engineer. Personal Interview. MITRE Corporation, Bedford MA, 18 June 1990.

17. Galorath Associates, Inc. SEER User's Manual. Marina del Ray CA: Galorath Associates, Inc., 1989.

18. Garvey, Paul R. and Frederic D. Powell. "Three Methods for Quantifying Software Development Uncertainty," Journal of Parametrics, VII: 76-91 (March 1987).

19. General Electric Company. PRICE S Reference Manual. Moorestown NJ: GE-Price Systems, 1990.

20. Gilb, Tom. "Software Cost Prediction Versus Software Cost Control," Journal of Parametrics, VII: 79-89 (March 1987).

21. Glass, Robert L. Software Soliloquies. New York: Computing Trends, 1981.

22. Gordon, Carl L., Charles R. Necco, and Nancy W. Tsai. "Toward a Standard Systems Development Life Cycle," Journal of Systems Management, 38: 24-27 (August 1987).

23. Greathouse, Richard M. and Kelly L. Shipley. "Current Research on Schedulers for Aerospace Industry Software." Presented at the Society of Aerospace Engineers Aerospace Atlantic, Wright-Patterson AFB OH, February 1990.

24. Gulezian, Ronald. "Fit, Error and Software Development Cost," Journal of Parametrics, IX: 62-72 (March 1989).

25. Hollander, Myles and Douglas A. Wolfe. Nonparametric Statistical Methods. New York: John Wiley & Sons, 1973.

26. IIT Research Institute. Test Case Study: Estimating the Cost of Ada Software Development. April 1989.

27. Jennings, Maj Kenneth R. Class notes in ORSC 542, Management and Behavior of Organizations, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, Jul - Sep 1989.

28. Jensen, Randall W. "An Improved Macrolevel Software Development Resource Estimation Model," Fourteenth Asimolar Conference on Circuits, Systems and Computers, Institute of Electrical and Electronic Engineers, New York: 1981.

29. Kile, Raymond L. REVIC User's Manual. Unpublished. 14 July 1989.

30. Martin Marietta Denver Aerospace Corporation. SASET User's Guide. Denver CO: Martin Marietta Corp., 1990.

31. Martin, Rick. "Evaluation of Current Software Costing Tools," ACM SIGSOFT Software Engineering Notes, 13: 49-51 (July 1988).

32. McGough, Keith. "Cost Estimation in Software Engineering," Journal of Parametrics, VII: 5-9 (June 1987).

33. Metzger, Philip W. Managing a Programming Project. Englewood Cliffs NJ: Prentice-Hall Inc., 1981.

34. Newbold, Paul. Statistics for Business and Economics. Englewood Cliffs NJ: Prentice-Hall Inc., 1988.

35. Parikh, Girish. Programmer Productivity. Reston VA: Reston Publishing Co., 1984.

36. Quantitative Software Management, Inc. SLIM Software Life Cycle Management. McLean VA: QSM, Inc., 1987.

37. Rampton, Judy Crockett. "Software Development Staffing Plans - From Estimate to Reality," Journal of Parametrics, VII: 51-58 (Dec 1987).

38. Reese, Richard M. and Jim Tamulevicz. "Software Sizing Methodologies," Journal of Parametrics, VII: 35-54 (June 1987).

39. Royce, Winston W. "Managing the Development of Large Software Systems," Proceedings of IEEE WESCON: 1-9 (1970).

40. Sierevelt, Hank. "Observations on Software Models," Journal of Parametrics, VI: 51-74 (December 1986).

41. Simpson, W. Dwain. New Technologies in Software Project Management. New York: John Wiley & Sons, Inc., 1987.

42. Software Productivity Research, Inc. User Guide SPQR/20. Cambridge MA: Software Productivity Research, Inc., 1989.

43. Weinberg, Gerald M. The Psychology of Computer Programming. New York: Van Norstrand Reinhold Co., 1971.

VITA

Captain Bryan A. Daly received the degree of Bachelor of Science from the United States Air Force Academy in 1982. His major concentration was in Management. He was commissioned on 2 June 1982 as a Second Lieutenant in the USAF. He first served as a Cost/Schedule Control Systems Criteria (C/SCSC) Surveillance Monitor at the Air Force Plant Representative Office at General Electric in Evendale, Ohio. He next was stationed at the Electronic Systems Division (ESD), working for one year as a C/SCSC Team Chief in the Cost Analysis section. He then worked for three years as a Cost Analyst in the Advanced Technology System Program Office at ESD, before entering the school of Systems and Logistics, Graduate Studies in Cost Analysis, Air Force Institute of Technology, in July 1989.

Permanent Address: 1 Elm Street

Newburgh, NY 12550

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1990	Master's Thesis

4. TITLE AND SUBTITLE

A COMPARISON OF SOFTWARE SCHEDULE ESTIMATORS

5. FUNDING NUMBERS

6. AUTHOR(S)

Bryan A. Daly, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GCA/LSQ/90S-1

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Accurate schedule estimation in software development programs is important because schedule is a major determinant of cost. Further, as a greater percentage of weapon system cost is taken by software, there is a greater need for knowledge in this area. In order to verify the accuracy of schedule prediction for software development obtainable today, this effort examined five commercially available software cost/schedule estimators. The estimated results were analyzed for their accuracy in predicting the actual schedules experienced on the projects. The models analyzed were REVIC, PRICE-S, System-4, SPQR/20, and SEER.

14. SUBJECT TERMS

Cost, Schedule, Software, Models

15. NUMBER OF PAGES
106

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL